

STUDY OF A GLOBAL SEARCH ALGORITHM FOR OPTIMAL CONTROL

Elwood C. Stewart, William P. Kavanaugh,

and David H. Brocker

Research Scientists

Ames Research Center, NASA

Moffett Field, California

SUMMARY

In this paper a feasible method of implementing the Maximum Principle is given based on an adaptive random search algorithm which utilizes direct measurements of the boundary cost-function hyper-surfaces. No restrictions are placed on the continuity of the surfaces or the number of valleys. The adaption enhances convergence by varying the mean and variance of a probability distribution as a function of the past performance. The algorithm has both local and global search properties so that "hanging up" in local valleys is avoided.

A hybrid computer implementation of the algorithm is discussed. The usefulness of the algorithm is investigated experimentally for a fifth-order nonlinear minimum-fuel orbit-transfer problem. Convergence is generally obtained within several thousand iterations (one to two minutes). Details are given on the manner in which the system iterates,

GPO PRICE \$
CFSTI PRICE(S) \$
Hard copy (HC) 3.00
Microfiche (MF) .65-

653 July 65

N 68-25436

(ACCESSION NUMBER) 77 (THRU) 1 (CODE) 10 (CATEGORY)
NASA-TMX-60529
(NASA CR OR TX OR AD NUMBER)

typical solutions obtained for a wide range of situations, and the convergence properties of several variations of the basic algorithm. Cross sections through the boundary hyper-surfaces reveal the striking irregularities which high-order systems can have and, hence, demonstrate the effectiveness of the adaptive random search approach in coping with them.

INTRODUCTION

The Pontryagin Maximum Principle is an exceedingly elegant and powerful theory for determining the optimal control of dynamic systems describable by nonlinear differential equations with bounded control. Although there has been a flurry of activity for several years in its application to low-order systems, there have been few applications to high-order systems. For high-order systems the Maximum Principle yields much information about the nature of the solution, but the actual solution is generally difficult to obtain. The reason for this is that a difficult mixed boundary-value problem is invariably encountered, one in which the known boundary conditions are divided between initial and terminal values. Furthermore, the mapping from the initial to terminal boundary values is not generally explicitly known.

One possible approach to the boundary-value problem is based on some form of the gradient method. However, there are several disadvantages in this approach: only local properties are utilized, only one local minimum is implicitly assumed, and the gradient is not analytically available. These difficulties were shown in reference 1 to lead to convergence problems. Thus the approach is of limited value when the surface being studied is multi-peaked, discontinuous, or very flat in certain regions. Random search methods would seem to be promising in overcoming some of the above deficiencies. Such methods have been used mostly in connection with direct search problems (refs. 2 and 3). However, in a recent study (ref. 4) a fixed-step, random-sign-type search was used to implement the Maximum Principle, and application was made to linear second- and third-order systems.

In this paper we will investigate a more general approach based on an adaptive random search method for solving the two-point boundary-value problem involved in the Maximum Principle. The search has both local and global properties. The method does not depend on the continuity of the surface being searched or the number of minima, and does not require that an analytical expression for the surface be known. In the sections to follow, the adaptive random search will be described, the hybrid computer realization discussed, and the results of

an application of the method to a fifth-order nonlinear orbit-transfer problem presented.

THE ADAPTIVE RANDOM SEARCH ALGORITHM

General Approach

The class of problems considered here are restricted to those for which the Maximum Principle is applicable. Familiarity with the Maximum Principle is assumed (refs. 5, 6, 7). Thus, we will not review the derivation, the theorems, the assumptions involved, or the boundary-value theory. For purposes of notation, however, a few remarks are appropriate. The system to be controlled is defined by the vector equation.

$$\dot{x} = f(x, u, t) \quad (1)$$

where $x = (x_1, x_2, \dots, x_n)$, $u = (u_1, u_2, \dots, u_m)$, and $u \in U$, the control region. Interest will center on fixed-time problems because of its convenience in computer operation. It will be desired to take the system from a given state $x(0)$ to a final target set S so as to minimize the generalized cost function

$$C = \sum_{i=0}^n \alpha_i x_i(T) \quad (2)$$

where $x_0(t)$ is the auxiliary state associated with the quantity to be minimized (ref. 7). Some of the possible target sets of interest are (a) $S \in R_n$, (b) $S \subset R_n$, and (c) $S = x_f \in R_n$, corresponding to a free point, a subset of the whole space R_n , and a fixed point. The solution to the problem invariably requires the solution to the set of equations

$$\left. \begin{aligned} u &= u(x, p, t) \\ \dot{x} &= f(x, u, t) \\ \dot{p} &= g(p, x, u, t) \end{aligned} \right\} \quad (3)$$

where the known boundary values are divided between the initial and terminal values. The final boundary condition required on p is dependent on the desired final boundary condition on the states.

The general approach to be used here to obtain explicit solutions to the Maximum Principle is based on an adaptive random search procedure. A hybrid computer diagram of the approach is illustrated in figure 1. We will be concerned with the fixed time interval 0 to T; however, it is trivial to vary this time in the computer implementation. The left half of the figure, indicated by the analog computation, is an implementation of the set of equations (3).

Fig. 1

The right half, indicated by the digital computation, is the algorithm which will enable the boundary conditions involved in the Maximum Principle to be satisfied.

Let us discuss the algorithm in detail. The basic notion in the random search algorithm is to select the initial condition vector for the adjoint equations from a noise source, in this case a gaussian distribution with fixed mean m and fixed standard deviation σ . Conceptually, the mean and standard deviation can be separated as indicated in the figure by adding the mean m to the output of a gaussian noise source with standard deviation σ and zero mean. Thus, on any k th iteration the vector p^k is

$$p^k = m^k + \xi^k \quad (4)$$

The gaussian source generates a purely random, gaussian, n -dimensional vector sequence $\{\xi^k\}$ with zero mean and independent components:

$$\left. \begin{aligned} m &\equiv E(\xi^k) = 0 \\ \text{cov}(\xi^j, \xi^k) &= Q \quad \text{if } j = k \\ &0 \quad \text{if } j \neq k \end{aligned} \right\} \quad (5)$$

where $\xi = (\xi_1, \xi_2, \dots, \xi_n)$

$$Q = \sigma^2 I$$

and I is the identity matrix.

Continuing around the loop in figure 1, the value of p^k as determined by equation (4) becomes $p(0)$, the initial condition for the adjoint equation. Since the value of $p(0)$ together with the given $x(0)$ is sufficient to define a

solution of the set (3), these equations, as represented by the left half of figure 1, can be integrated to the terminal time T . The final state $x^k(T)$ actually achieved will generally fail to satisfy the desired terminal state as defined by the target set. Similarly, the final values of the adjoint variables $p^k(T)$ will fail to satisfy the boundary conditions required depending on the target set. For this reason we will introduce a function to measure the difference between the actual boundary values and the desired boundary values. The deficiencies pointed out in reference 8 of the scalar-valued metric will be avoided. We will partially order systems by a vector-valued metric. Since there will generally be three distinct types of boundary conditions to satisfy at the terminal time, we will take the vector metric of the form

$$J = (J_D, J_V, J_P) \quad (6)$$

where the components refer to displacement, velocity, and adjoint variables, respectively. For two systems S and S' we will say $S > S'$ if and only if $J > J'$, that is, $J_D > J_D'$, $J_V > J_V'$, and $J_P > J_P'$. With this concept one is concerned with choosing from a certain set of systems a noninferior system rather than an optimum system. This corresponds more closely with a realistic objective as pointed out in reference 8.

Assume now that the search is purely random with no adaptive characteristics. In this case the dashed lines indicated in figure 1 would be absent. Then as a result of the random vector sequence $\{t^k\}$, the vector sequence $\{J^k\}$ is generated. Since the search is purely random, the algorithm logic decides at each iteration whether the value of J has been reduced to zero.

Satisfying the boundary conditions is slightly more involved than the above because in an experimental study it is not likely the boundary conditions for x and p will ever be precisely achieved. For this reason we will enlarge the target set by some small amount and allow the system to terminate at any point in the enlarged set. This view is more realistic, since there is no reason to demand that a practical system satisfy the boundary condition exactly. To accomplish this in the random search method, we will require of the vector metric

$$J < \epsilon \quad (7)$$

where

$$\epsilon = (\epsilon_D, \epsilon_V, \epsilon_P)$$

The values of ϵ_D and ϵ_V are dictated by how closely it is desired that the system states approach the desired target set. The value of ϵ_P is more difficult to choose because of its lack of physical interpretation. Fortunately, the hybrid computer approach makes it easy to choose a sufficiently small value by experimentally observing the sensitivity

of the solutions to small changes in ϵ_p . In an experimental study to be described later, the solutions were quite insensitive to variations in ϵ_p over a wide range.

The Adaptive Algorithm

The pure random search described in the preceding section is generally unsatisfactory because of the excessively long convergence times, as will be seen in a later example. We attempt to improve the convergence properties by making the system adaptive by varying the mean m and standard deviation σ as a function of the system's past performance. Since performance is determined by the input and output sequences, $\{\xi^k\}$ and $\{J^k\}$, we will make the mean and standard deviation adaptive of the form

$$\left. \begin{aligned} m^{k+1} &= f_m^{k+1}(\xi^1, \xi^2, \dots, \xi^k, J^0, J^1, \dots, J^k) \\ \sigma^{k+1} &= f_\sigma^{k+1}(\xi^1, \xi^2, \dots, \xi^k, J^0, J^1, \dots, J^k) \end{aligned} \right\} \quad (8)$$

Some reasonable and easily implemented forms will be discussed in the following paragraphs. A fundamental notion in the algorithm will be that of a "success" or "failure" defined by a function of the cost J^k on any k th iteration and the smallest preceding cost J^l obtained on the l th iteration. The most frequently used definitions were simply:

$$\left. \begin{aligned} \text{success:} & \quad J^l - J^k > 0 \\ \text{failure:} & \quad J^l - J^k \leq 0 \end{aligned} \right\} \quad (9)$$

Obviously a successful iteration is both necessary and sufficient for the system to be noninferior. It might be noted that when any component of J^k is less than its corresponding ϵ component, we will not require further reduction for a success as long as it remains less than ϵ .

Implementation of the adaptive part of the algorithm is illustrated in figure 1 by the dotted lines and the algorithm logic block. The algorithm logic block performs the computations in the above equations by utilizing: (1) the p^k information from the memory M_p , and (2) the system performance information, J^k , from the memory M_j . The output of the algorithm logic block is then the mean and standard deviation of the distribution for the next iteration as indicated.

The rationale behind the particular adaptive law to be used here is based on the notion of a creeping, expanding, and contracting search. The creeping character of the search is provided by varying the mean of the distribution so as to equal the initial condition of the adjoint vector on the last successful iteration. The expansion and contraction character of the search is provided by varying the standard deviation such that the search is localized when successful but gradually expanded when not successful. These characteristics give the algorithm some useful search properties.

The adaptive law for the mean value of the distribution was taken to be:

$$m^{k+1} = \begin{cases} p^k & \text{if } J^k < J^{k-1} \\ m^k & \text{if } J^k \geq J^{k-1} \end{cases} \quad (10)$$

where the initial values are $m^1 = 0$ and J^0 is the value based on the initial values $x(0)$. That this law is of the form (8) can be seen by combining (4) and (10) sequentially. Thus, the first few terms of the sequence are

$$\begin{aligned} m^1 &= 0 \\ m^2 &= \begin{cases} \xi^1 & \text{if } J^1 < J^0 \\ 0 & \text{if } J^1 > J^0 \end{cases} = f_m^2(\xi^1, J^0, J^1) \\ m^3 &= \begin{cases} \xi^1 + \xi^2 & \text{if } J^2 < J^1 < J^0 \\ \xi^1 & \text{if } J^2 > J^1 < J^0 \\ \xi^2 & \text{if } J^1 > J^0, J^2 < J^0 \\ 0 & \text{if } J^2, J^1 > J^0 \end{cases} = f_m^3(\xi^1, \xi^2, J^0, J^1, J^2) \end{aligned} \quad (11)$$

The adaptive law for the standard deviation of the distribution, σ , was implemented so as to expand the size of the search depending on performance. Let J^l be the metric obtained on the last successful iteration. Then the law for the standard deviation on any k th iteration is

$$\left. \begin{aligned} \sigma^k &= \sigma_1 && \text{if } k - q < l \\ &= \sigma_2 && \text{if } J^k, \dots, J^{k-q} > J^l \\ &\vdots \\ &= \sigma_\gamma && \text{if } J^k, \dots, J^{k-(\gamma-1)q} > J^l \end{aligned} \right\} \quad (12)$$

where the σ_i 's are constants such that $\sigma_1 < \sigma_2 < \dots < \sigma_\gamma$.

In other words, σ_1 is used if there was a success in the last q iterations, σ_2 if there was no success in the last q iterations, σ_3 if no success in the last $2q$ iterations, etc.

These equations are, of course, a simple form of equations (8).

Occasionally, depending on the difficulty of the particular problem being studied, no success will be achieved in the γq iterations. For this reason, we repeat the search in equations (12) a number of times C (usually twice), and then reinitialize the entire search if no success is achieved.

The parameters which are free to choose are q , γ , and the σ_i 's; in the example to be discussed later values of $q = 100$ and $\gamma = 10$ were chosen and they do not seem to be critical.

The choice of σ values is more important because it affects both the smallness and the largeness of the search. However, it is not difficult to choose reasonable values. A reasonable lowest value, σ_1 , can be determined by observing on the display system (yet to be described) the metric J on every iteration, and choosing a value small enough so that the J generally varies only slightly. For the orbit-transfer problem discussed later, $\sigma_1 = 0.1$ volt proved satisfactory.

The upper value is chosen to cover some sizable portion of the entire space; a value of 10 volts seemed reasonable for the 100 volt space available on the analog computer. In between, the steps are a geometric progression to enable the search to expand rapidly.

Several additional adaptive strategies were incorporated into the algorithm to provide better convergence properties and greater versatility in certain situations. They are itemized by the following:

1. Single-step strategy -- This strategy is intended to take advantage of favorable local properties of the surface. It is based on the notion that the step following a successful step k should not be random but deterministic in the same direction and of the same amount. That is, p^{k+1} is defined by

$$p^{k+1} = m^{k+1} + \xi^k \quad \text{if } J^{k-1} - J^k > 0 \quad (13)$$

The cost in time, one iteration, is insignificant, and the benefits are substantial as will be seen in a later example.

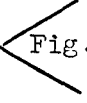
2. Threshold strategy -- In this strategy the requirement for a success is modified so that the difference between the cost for the k th iteration and that for the last successful iteration (i.e., the left side of equations (9)) must exceed some threshold value dependent on the cost function. We take the case in which the threshold is simply ηJ^l where $0 < \eta < 1$. Thus, a success is defined by

$$J^l - J^k > \eta J^l \quad (14)$$

This strategy has the desirable characteristic that smaller improvements are required as the minimum is approached.

3. Localized end-search strategy -- This end-search strategy is intended to localize the search when in the immediate neighborhood of the minimum. This is done by reducing the standard deviation in equations (12) so that σ varies from $\alpha\sigma_1$ to $\alpha\sigma_\gamma$ when $J^k < \beta$, where $0 < \alpha < 1$, and β is some small value generally on the order of 2ϵ . This strategy was used sparingly since it was beneficial in reducing convergence time only under certain conditions. Although a gradient method could be used in this final phase, the random search method accomplishes the same objective without an implementation change.
4. Initial search -- It was found advantageous to control the distribution from which the first successful adjoint vector $p(0)$ is obtained. This was done by starting the search with a square distribution of variable width, $-W$ to $+W$. When a success is obtained the search continues with a gaussian distribution as described above. The rationale behind the square distribution is that since no information is available on the best starting value, the weighting within the

limits should be equal. The size of W is a compromise: It must be large enough to include possible solutions but not so large that the volume being searched is excessive. The choice of W will be dependent, of course, on the particular problem; it is not difficult to choose experimentally a reasonable value. Values of 15 to 30 volts are typical for the later example problem.

It is clear the behavior of the adaptive algorithm is different from the pure random or nonadaptive search described before. Now the vector metric J which measures the disparity between desired and actual terminal boundary conditions is sequentially reduced rather than being reduced in one iteration. The way in which this occurs is illustrated in figure 2, where  Fig. 2 a representative surface is given in only two dimensions. Due to its adaptive character, after any success the mean of the distribution moves to the last successful p . At this point the search starts locally and increases in a geometric progression until the next success is obtained. In this way, the successive values of J may jump from one valley to another as indicated by the numbered points until a J less than the required ϵ is reached. The algorithm logic block decides when this condition occurs.

The virtues of the random search approach are clear. It has desirable local and global search properties so that "hanging

up" in local valleys as with gradient methods is avoided. Further, it will not matter if the surface is discontinuous or how many peaks or valleys there are. The main question is, of course, the convergence time. This is best studied experimentally on an example to be discussed in a later section.

IMPLEMENTATION

Considerations in the Choice of Computer System

The hybrid computer proved to be the most feasible way in which to implement the random search algorithm. The reasons for this and a comparison with the alternatives are worth discussing.

In general there are three basic computational techniques available to the experimenter for the implementation: analog, digital, or a combination of the two, hybrid. Of prime importance in the selection of the computer system is a consideration of the number of iterations required to find a solution, and this number is not known a priori. For a second-order system, pilot studies indicated something on the order of 10^2 iterations to obtain a solution. Since the volume of the space increases so rapidly with the order of the system, one might expect several thousand iterations to be necessary for an increase in system order to perhaps five or six. Thus, we see that the time per iteration

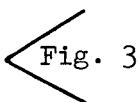
will be of critical importance and will largely dictate the means for implementing the search algorithm.

Each iteration can be divided into two steps:

(1) integration of the equations of motion on the interval $[0, T]$, and (2) execution of the algorithm. For the orbit-transfer problem to be discussed later, an IBM 709⁴ machine requires 1-10 seconds to perform the integration in step 1. An analog computer, however, performs the same integration in 1-10 milliseconds with an accuracy of about 5 percent relative to the digital machine solutions. Thus for this specific example, the analog computer is about 10^3 faster than the digital computer in performing the integration in step 1. The second step is best accomplished digitally. The time to perform the second step on a digital computer of speed comparable to the IBM 709⁴ is the same order of magnitude as the time required for the analog to perform step 1. Therefore, a great saving in computer time can be realized over a completely digital simulation by a hybrid approach. It is worth noting that an alternative approach was investigated utilizing pseudo-hybrid techniques, that is, an analog computer and something less than a digital computer. However, our experience shows that inaccuracies, limited storage, and limited flexibilities in logical operations seriously limit the feasibility of this approach.

The Hybrid System

In the hybrid implementation the analog computer was delegated the task of solving the state, adjoint, and control equations, as given in equations (3). It also served as the point at which the operator exercised manual control over the hybrid system. The digital computer was required to calculate the metric, provide storage, implement the algorithm, generate the initial conditions for the adjoint equations, and finally, oversee the sequencing of events of the iterate cycle.

Figure 3 is a hardware diagram of the hybrid system used.  Fig. 3
Shown are the two basic elements of the simulation, the analog and digital computers along with their coupling system, and peripherals. The coupling system is comprised of two distinct parts: (a) the Linkage System and (b) the Control Interface System. A discussion of the hardware used in these subsystems is given in the four sections to follow. A final section discusses the sequencing of events through the subsystems during one iteration cycle in order to better describe the functioning of the hybrid system as a whole.

A. Digital Computer -- The digital computer used in the optimization program was an Electronic Associates, Inc. (EAI) model 8400. It is a medium-sized, high-speed computer which is designed to operate in a hybrid atmosphere. The particular machine used has 16,000 words of core memory

with 32 bits per word. Memory cycle time is two microseconds. The machine uses parallel operation for maximum speed. Floating point operations are hardware implemented. Programming languages available include a MACRO ASSEMBLY language and FORTRAN IV. The optimization program was coded in MACRO ASSEMBLY in order to keep the execution time to a minimum. The instruction repertoire includes special commands by which discrete signals can be sent to or received from the external world. External interrupts are provided which can trap the computer to a specific cell in memory. In an example to be discussed later the optimization program utilized about 8,000 words of storage. Of this about 1,000 comprise the actual optimization executive program, the remaining 7,000 being used for subroutines, monitor and on-line debugging and program modification routines.

The peripherals of the digital computer include magnetic tapes, card reader, and printer. The two former devices were used for program input and storage while the latter device was used for data logging.

B. Analog Computer -- The analog hardware consisted of an Electronic Associates 231R-V analog computer. The state equations, adjoint equations, and the control logic were programmed in standard fashion. Consequently, analog schematics were thought not essential in this paper.

The analog computer serves as the point at which mode control of the hybrid computer is accomplished. By manual selection of switches either of two modes can be commanded: (1) In the "search" mode the analog computer operates in a high-speed repetitive manner. Such operation is accomplished by controlling the mode of the individual integrators with an appropriate discrete signal. This signal is a two-level signal which is generated on the control interface in conjunction with the digital computer and, depending on the level, holds an integrator in either "operate" or "initial condition" mode. (2) In the "reset" mode, the integrators are placed in their initial-condition mode and held there.

Particular equipment worth pointing out are the track-store units, D/A switches, and comparators with which the control logic was implemented. At the end of the operate period T , the digital computer reads a number of variables essential to its functioning. Because of the high repetitive speeds used, the value of a variable could change considerably between the end time T and a later time when it is actually converted. Thus, track-store units were used to hold the variables at their respective values at time T until the digital read all of the values. The control logic requires on-off type switching and the high-speed electronic comparators and electronic switches were

quite necessary for proper operation. Nonlinear operations such as multiply and divide provided no particular difficulties, in their normal operation, although a square root operation did require the use of a diode function generator.

For continuous type output, a display console was connected to the analog computer to provide visual readout of variables. The display contained a cathode ray tube (CRT) which could simultaneously display up to four channels, and enabled photographic records to be taken of the display quantities. The display was extremely helpful in determining if the algorithm was functioning properly. Other continuous analog data useful for determining proper functioning were the p^l and J^l ; these could be recorded on a pen recorder since their rate of change was low.

C. Control Interface -- The control interface between the analog and digital system is an Electronic Associates, Inc. DOS 350 (see fig. 3). It is through this unit that the iteration process is controlled. An important task allocated to this subsystem is the operate-time control. This function is implemented through the use of a counter and is the key element in the control of all timing in the hybrid simulation. The counter is driven from a high-frequency source in the interface system allowing for a very high degree of resolution in the simulated operate-time. The counter is constructed by patching modular blocks which can be combined

to give a wide range of simulated operate-time. Specific times within this range are selected with thumb-wheel switches. Also, the interface allows the digital computer to use any conditions in the analog computer which can be represented by discrete variables (binary levels) and to send discrete signals to the analog system to be used as control levels or indicators. An example of the former would be the hybrid system mode control which merely amounted to the operator depressing the "reset" or "search" switch on the analog console. This action sets a binary level which is then sensed by the digital computer. An example of the latter situation is when the digital sends the operate command to the operate-time counter. The interface system allows patching of Boolean functions. Hence, some of the logic operations required for timing pulses, event signals, and other like operations were very effectively programmed on it.

D. Linkage System -- The linkage system which is shown in figure 3 is that part of the system which houses the conversion equipment, the A/D and D/A converters. It is through here that all of the data passes between the analog and digital portions of the simulation. The linkage system is controlled by command from the digital computer.

Input to the digital computer is through the A/D converter which has preceding it a channel selection device

or multiplexer to select the analog channel which is to be converted. Conversions were done sequentially through the analog channels at a maximum rate of 80,000 samples per second from channel to channel.

Output to the analog used the D/A converters with each data channel having its own conversion unit. The maximum conversion rate of the D/A's used is 250,000 conversions per second.

E. Sequencing of Events During One Iteration -- The sequencing of events during one iteration cycle are depicted in figure 4. The instants of time t_1, t_2, \dots, t_5 shown in this figure are considered fixed relative to each other, and t_1 is conveniently regarded to be the start of the iteration cycle. We will consider the cycle to begin at t_1 with the analog integrators in an operate mode. As discussed previously, the elapsed time $(t_2 - t_1)$ is controlled by a counter on the interface system. At t_2 an interrupt pulse is generated on the control interface which is sent to the digital computer signaling it to commence its operations. Simultaneously, the pulse is sent to the analog to instruct the track-store units to hold their respective values which they possessed at time t_2 . During the interval $(t_3 - t_2)$ the digital computer reads these analog variables with the A/D converter. At t_3 the digital sends a pulse via the




Fig. 4

interface to the analog console which commands the integrators to an initial condition mode. At t_4 , when the data required by the algorithm has been generated the D/A converters send these values to the appropriate points in the analog portion of the simulation. The digital machine allows enough time for the transients to settle in the initial condition circuits of the analog before sending a command at t_5 that places the integrators in an operate mode and starts the counter. Since t_5 and t_1 are the same event, we merely repeat the above sequence for repetitive operation.

Some specific numerical values might be of interest. The total iterate time ($t_5 - t_1$) is primarily composed of two parts: (1) ($t_2 - t_1$) which in a later example problem was scaled in the simulation to 2.5 milliseconds, and (2) ($t_5 - t_2$), which was primarily determined by the speed of the digital machine in computing, converting, and generating random numbers; this latter period was on the order of 7.5 milliseconds. Thus, the total iterate time for the above situation is on the order of 10 milliseconds (or 100 iterations per second). This figure is dependent on the control problem chosen and the exact form of the algorithm implemented.

The Algorithm Flow Graph

A program flow graph which shows the operation of the iteration process is displayed in figure 5. This basically

is the flow graph of the algorithm and the iteration control sequences utilized by the hybrid system. Note the inclusion of the event times t_1, t_2, \dots discussed earlier in connection with figure 4. The program is continuously recycling in a high-speed repetitive fashion.

There are three basic loops which correspond to the three system modes in the optimization program: a reset loop, a search loop, and an end-state loop. The reset loop is for the purpose of initializing the program. The search loop is that portion of the program which uses the algorithm to search for a solution to the problem. The end-state loop is entered by the digital program when a solution is found, and is used for the generation of graphic displays. The operator manually selects the search or reset mode as discussed in the section dealing with the analog computer.

A. Reset Loop -- The reset loop is the portion of the repetitive operation cycle which initializes the program and prepares it for the search mode. When the system is in reset mode, the integrators of the analog computer remain in their initial condition state. The flow of the reset cycle is shown in figure 6 where that particular loop is emphasized by line weight, figure 6 being the same as figure 5 otherwise. It is while the system is in the reset mode that the digital program is first entered, the point of entrance being designated by START in the figure. The first

Fig. 6

operation performed is that of initialization. It is during this process that all the program variables are set to their initial states. Also, the line printer is initialized and the data header printed. Once complete, the interrupt line is enabled and the analog computer signaled that an iteration cycle is to begin. It is at this point that the counter which controls the operate time of the analog is started. The digital computer then halts and waits for an interrupt to signal that the period T has ended.

When the interrupt occurs, the digital program proceeds to the next operation where the values of the states $x^k(T)$ and $p^k(T)$ of the analog computer are read by the A/D converter. Once the inputs to the digital computer have been read, a pulse is sent to the analog computer to place the integrators in their initial condition mode. Since the integrators are already in initial condition state due to the reset mode, this pulse has no effect but it is needed later when the system is in the search mode or the end-state mode where the integrators are not so constrained.

The next operation performed is that of calculating the metric, J^k , based on the samples of the state obtained in the last block. This value is peculiar to the reset mode and, therefore, is designated J^0 and saved as such. Once

the metric is calculated, the digital computer tests the system mode and branches to that portion of the program which is exclusive to the reset loop.

The program now initializes the algorithm by setting $m^k = p^l(0) = 0$ and $J^l = J^0$. The values of the components of $p^k(0)$ are then generated using a uniformly distributed noise source, the space of the noise being $[-W, W]$. Note that in reset mode $p^k(0) = \xi^k$.

At this point the reset loop returns to the mainstream of the program by entering the output portion of a cycle. It is here that all of the quantities required at the analog computer are converted to analog form by the D/A converters. The data sent to the analog include $p^k(0)$, $p^l(0)$, J^l , and J^k . Note that $p^k(0)$ is required by the analog computer whereas the others are displayed to determine if the algorithm is functioning properly.

Finally the digital program enables the interrupt line and signals the analog computer and control interface to begin another cycle. As before, the program now goes into a halted state waiting for the interval T to pass. Cycling in the reset loop continues until the operator is ready to begin a search cycle and does so by putting the hybrid system in the search mode.

B. The Search Loop -- It is in the search mode that the algorithm is used to seek an optimal solution to the control

problem implemented. Search mode is selected by the operator at any time and is considered to begin on the first iteration after his doing so.

During the first iteration of search mode, the analog computer solves the system, adjoint and control equations using the $p(0)$'s established in the last cycle through the reset loop. The digital computer then receives the time T interrupt signal and proceeds to read the states $x^k(T)$ and $p^k(T)$ as shown in figure 7. Once read, the integrators are set back to initial condition where they will await the next $p(0)$ to be generated by the algorithm. The next operation computes the metric as was done in the reset loop; only this time the J^k is based on the actual solution of the system equations at time T .

Fig. 7

The system mode test is next and results in a branch to that portion of the program that solves the algorithm and generates the $p(0)$'s based on the adaptation principles. The search branch begins with a test to determine if the system is in end state. Since this is the first time through the search loop, end-state condition cannot have been established so the program proceeds to the operation which tests the metric. Here all of the components of the metric are tested according to equations (9) and the condition of a success or failure ascertained as discussed before.

At this point in the discussion we shall take the results of the metric test to be a success so that that portion of the search loop can be examined. The first operation in this segment up-dates the memory of the algorithm by setting the last good values of the $p(0)$'s to $p^k(0)$, the vector which gave this success, and by setting the last good metric J^l to J^k . Next a test is made on the metric to see if it meets the requirement of a solution, i.e., $J^l < \epsilon$. Assume for the moment that it does not and that the program proceeds out the lower branch of the test. The program then generates the next try for the adjoint initial conditions on the basis of the single-step strategy. Following the establishment of $p^{k+1}(0)$ the program tests J^k to determine if the end-search strategy should be employed by comparing J^l with δ . If J^l is less, the noise standard deviation sequence is reduced through multiplication by the constant α ; if J^l is not less than δ , the standard deviation is unaltered. Then after setting σ to the starting value σ_1 , the program execution returns to that part which is common to all loops, the output portion. The program then starts the next iteration exactly as it does in the reset loop.

If the test of the metric had resulted in a failure, the program would have branched to the right after the metric test shown in the flow graph. Here the primary function is to generate a new set of adjoint initial conditions using the noise generator as discussed in the description of the

adaptive algorithm. Before the actual generation of $p^{k+1}(0)$ can take place, however, certain of the algorithm parameters must be examined. These parameters control the maximum time at a given σ level, the changing of σ levels, and whether or not to start the search anew. First, a test is performed to find out if q consecutive trials have resulted in failures. If not, σ remains fixed and the program executes the jump ahead shown; if true, σ is incremented to the next value in the sequence $(\sigma_1, \dots, \sigma_\gamma)$. Next is a test to determine if $\sigma > \sigma_\gamma$. If not, a jump ahead is executed. If it is, then a test is made to determine if the sequence has been gone through C times. If so, the program will reinitialize itself by entering the reset loop for a new start. This does not put the hybrid system in reset state but only restarts the algorithm. If the program did not reinitialize, then σ is set to σ_1 , to prepare it for another cycle through the sequence.

This brings the execution of the program to the generation of the noise vector ξ^{k+1} (where it would have been if there were less than q consecutive failures or if σ was not greater than σ_γ). The distribution of the noise generator is uniform over the space $[-W, W]$ if no success has been achieved since first entering the search loop. Otherwise the distribution is gaussian with zero mean. Next, the new adjoint initial conditions are obtained by adding ξ^{k+1}

to the last set of adjoint initial conditions which produced a success, and the program moves to the output section.

The search mode continues reducing the metric by the selection of $p(0)$ until a value is found which gives a solution to the simulated control problem. This solution is sensed at the point in the program where J^2 is compared with ϵ . This test results in a branch to the left where the first operation is to set up the end-state loop condition. Next, the data representing a solution are logged on the printer and the new adjoint initial conditions set to the value which gave the solution. The program then goes to output with the system set to perform the end-state loop.

C. End-State Loop -- The end-state loop is shown in figure 8 by the line weight emphasis. In end state the integrators still sequence through their initial condition or operate modes just as in the search mode. However, $p(0)$ remains fixed at that value which produced the solution. In this manner it is possible to observe the optimal solutions found by the search algorithm on the CRT display described earlier.

Fig. 8

AN EXAMPLE ORBIT-TRANSFER PROBLEM

In this section the usefulness of the random search algorithm in solving a moderately difficult problem will be discussed. An orbit-transfer problem was selected because

it is high order, nonlinear, and is the type of problem for which optimization is appropriate. That is, since such large amounts of fuel are involved in effecting an orbit transfer, it would be profitable to minimize the required fuel. In the first four subsections, we will formulate the problem, show the equations which result from an application of the Maximum Principle, and discuss the boundary conditions and the associated metric; in the last four subsections, we will present some experimental results on the characteristics of the random search algorithm and on the orbit-transfer problem.

Formulation

The particular orbit-transfer problem considered was a transfer from an earth-moon trajectory to a circular orbit around the moon. The physical situation is illustrated in figure 9 for the planar situation. The final desired orbit is a circle 190 km above the lunar surface, i.e., a circle of radius 1928 km. The midcourse corrections might have placed the vehicle on some typical orbits as shown in which pericynthian may or may not coincide with the desired final orbit. The equations of motion in the vicinity of the moon will be governed by two-body equations and the trajectories will be hyperbolas as indicated. For purposes of simulation, a good coordinate system to describe the motion is the rotating coordinate system shown in figure 9; the origin is on the circular orbit and its velocity is the same as for a particle in that circular orbit. This coordinate system

Fig. 9

is a desirable one to use because it subtracts out large constant values from the inertial coordinate system. Assuming the vehicle is to be controlled by gimbaling a thrusting engine in which the mass flow rate is used to vary the thrust, the exact equations of motion in the rotating coordinate system shown in figure 9 are:

$$\left. \begin{aligned} m(t)\ddot{X} - 2\omega m(t)\dot{Y} &= -\dot{m}(t)c \cos \alpha + m(t)\left(\omega^2 - \frac{\mu}{r^3}\right)X \\ &\quad + M(t)\left(\omega^2 - \frac{\mu}{r^3}\right)r_c \\ m(t)\ddot{Y} + 2\omega m(t)\dot{X} &= -\dot{m}(t)c \cos \beta + m(t)\left(\omega^2 - \frac{\mu}{r^3}\right)Y \end{aligned} \right\} \quad (15)$$

The gimbaling implies the constraint

$$\cos^2 \alpha + \cos^2 \beta = 1 \quad (16)$$

In the above equations $m(t)$ is the vehicle mass, ω is the angular velocity of the rotating coordinate system, c is the exhaust velocity, α and β are the angles between the thrust vector and the X and Y axes, respectively, r is the radius to the vehicle in the inertial coordinate system, r_c is the radius of the desired circular orbit, and μ is a gravitational constant. Typical values used were $m(0) = 39,096$ kg, $c = 3.1405$ km/sec, $r_c = 1928.68$ km, $\mu = 4,890$ km³/sec², $\dot{m}(t) = -31.07$ kg/sec, $\omega = 8.259 \times 10^{-4}$ r/sec.

Now if we let $x_1 = X$, $x_2 = \dot{X}$, $x_3 = Y$, $x_4 = \dot{Y}$, and $x_5 = m(t)$, the equations of motion can be put in the state vector form

$$\dot{x} = f(x, u)$$

where $x = (x_1, x_2, x_3, x_4, x_5)$ and $u = (u_1, u_2, u_3)$. In expanded form we have the set of five nonlinear differential equations:

$$\left. \begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= 2\omega x_4 + \frac{cu_1 u_3}{x_5} + \left(\omega^2 - \frac{\mu}{r^3}\right)x_1 + \left(\omega^2 - \frac{\mu}{r^3}\right)r_c \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= -2\omega x_2 + \frac{cu_2 u_3}{x_5} + \left(\omega^2 - \frac{\mu}{r^3}\right)x_3 \\ \dot{x}_5 &= -u_3 \end{aligned} \right\} \quad (17)$$

where $u_1 = \cos \alpha$, $u_2 = \cos \beta$, and $u_3 = -\dot{m}(t)$. It is worth noting with respect to simulation accuracy that by using the rotating coordinate system the terms involving r which are difficult to simulate accurately provide small corrections to the more dominant terms involving only the states in the rotating coordinate system. These correction terms cannot be ignored, however. A simpler model which ignores these terms was found to introduce significant errors in end states and fuel required.

The task to be accomplished is to minimize the fuel to go from a given position on the hyperbolic orbit to the circular orbit. Thus, the quantity of fuel used is introduced as the added coordinate:

$$x_0(t) = \int_0^t u_3(\tau) d\tau \quad (18)$$

and we can interpret the objective as the minimization of the terminal value $x_0(T)$. This additional coordinate

requires the corresponding differential equation

$$\dot{x}_0(t) = u_3(t) \quad (19)$$

to be adjoined to the set (17).

Maximum Principle Solution

Although application of the Maximum Principle to the present problem will not be given in detail here, the equations important to the hybrid computer implementation will be summarized. First are the adjoint equations which are fundamental to the development. The approximate adjoint equations can be shown to be

$$\left. \begin{aligned} \dot{p}_0 &= 0 \\ \dot{p}_1 &= -p_2 \left(\omega^2 - \frac{\mu}{r^3} \right) \\ \dot{p}_2 &= -p_1 + 2\omega p_4 \\ \dot{p}_3 &= -p_4 \left(\omega^2 - \frac{\mu}{r^3} \right) \\ \dot{p}_4 &= -p_3 - 2\omega p_2 \\ \dot{p}_5 &= \frac{c \|P\|}{x_5^2} u_3 \end{aligned} \right\} \quad (20)$$

Second are the equations for the optimal control vector which have been derived from the Maximum Principle:

$$\left. \begin{aligned} u_1 &= \frac{p_2}{\|P\|} \\ u_2 &= \frac{p_4}{\|P\|} \\ u_3 &= M \quad \text{if } v = \|P\| - \frac{x_5}{c} (p_5 + 1) > 0 \\ &0 \quad \text{otherwise} \end{aligned} \right\} \quad (21)$$

where $\|P\| = \sqrt{p_2^2 + p_4^2}$ and M is the magnitude of the maximum thrust. It is seen that the thrust magnitude is either on or off depending on the sign of the switching function v , and the thrust angles are continuous functions of the adjoint variables. To obtain an explicit solution, it is now necessary to solve simultaneously by means of the analog computer the equations (17) for x , equations (20) for p , and equations (21) for control u , subject to certain boundary conditions yet to be discussed. The auxiliary differential equations for $x_0(t)$ and $p_0(t)$ do not couple to the other equations; they can, therefore, be dropped in the implementation to be described later.

Boundary Conditions

The boundary conditions, as yet unspecified, are the heart of the random search method. The initial values for the state vector $x(0)$ are fixed at the given values while the initial values of the $p(0)$ vector are chosen by the algorithm. The desired terminal conditions can be specified

in a variety of ways depending on the target set chosen. On the one hand, the end points could be treated as fixed at some point such as the origin of the coordinate system. This is somewhat restrictive. On the other hand, the end points could be treated as variable by taking the target set to be the desired circular orbit. This approach would yield the best point in the target set but would require complicated transversality conditions to be satisfied. We will take a middle ground in the interests of not unduly complicating the problem. We will take the target set to be the desired circular orbit, but we will consider the end point fixed at whatever point in the target set is reached in the fixed time, T . The advantage of this specification is that the boundary conditions are somewhat simpler than if transversality conditions had been included. With these boundary conditions the vehicle will reach some point on the desired circular orbit but perhaps not the best point. However, as we will see in a later example, the fuels for the large majority of solutions were so close to the theoretical minimum fuel based on impulsive orbit transfer that it does not appear that satisfying transversality conditions could result in a better solution with lower fuel. This conclusion was found valid over the range for which the problem was studied.

The specific values for the terminal boundary conditions can be found from conventional theory and summarized as:

$$\left. \begin{array}{ll} x_1(T), x_2(T), x_3(T), x_4(T) & \text{fixed} \\ x_5(T) & \text{free} \\ p_1(T), p_2(T), p_3(T), p_4(T) & \text{free} \\ p_5(T) = 0 & \text{fixed} \end{array} \right\} \quad (22)$$

The Vector Metric

For purposes of satisfying the boundary conditions, the components of the vector metric J need to be specified. Since the target set is taken to be the circular orbit, a suitable displacement metric is

$$J_D = |r(t) - r_c| \quad (23)$$

where r_c is the radius of the desired circular orbit. As for the velocity metric, it is clear that for the vehicle to stay close to the desired circular orbit after the terminal time T , we would like: (a) the radial component of the inertial velocity, V_R , to be small, and (b) the tangential component, V_T , to be close to the velocity consistent with that for the circular orbit, i.e., $r_c\omega$. Thus, a reasonable metric which measures the errors in these velocity components is

$$J_V = \sqrt{(r_c\omega - V_T)^2 + V_R^2} \quad (24)$$

The quantities V_T and V_R are simple transformations of the states in the rotating coordinate system. The only adjoint variable which must be fixed at the terminal end is p_5 so

that we use simply

$$J_p = p_5^2(T) \quad (25)$$

The values to which the components of J must be reduced are somewhat dependent on the mission and accuracy requirements after the terminal time. For most of the study values were chosen to be

$$\left. \begin{aligned} \epsilon_D &= 8 \text{ km} \\ \epsilon_V &= 15 \text{ m/s} \\ \epsilon_P &= 0.5 \text{ volt} \end{aligned} \right\} \quad (26)$$

This velocity requirement reduces the terminal inertial velocity to less than 1% of the initial value. The adjoint variable requirement was readily found experimentally. As mentioned in the discussion following equation (7), values of ϵ_P several times this value were found to be satisfactory.

Behavior and Characteristics of the Algorithm

The implementation of the random search algorithm for the orbit-transfer problem followed closely the discussion in a previous section and no further details will be given here. With this implementation the algorithm was studied to determine some of its important properties and the effect of some of its possible variations. The results of such studies will be discussed in this section.

In order to study the algorithm, we will confine our interest to one particular situation of the orbit-transfer problem. This situation was chosen such that the vehicle is approaching the moon on the hyperbolic orbit number 2 as shown in figure 9. Pericynthian coincides with the desired final circular orbit, which is 190 km above the lunar surface. Rather arbitrarily the initial starting point was chosen as -37° , and the time allowed for the vehicle maneuver was taken to be 600 seconds. For comparison, the time to reach pericynthian with zero control is 534 seconds.

There are two displays which well illustrate the manner in which the system iterates and searches for a solution. The first display illustrated in figure 10 gives information about the successive improved iterations. In this figure is shown the successive improved values of the initial adjoint vector $p(0)$ and the corresponding decrease in the boundary cost functions J_D , J_V , and J_P . It is worth noting the creeping nature of the $p_3(0)$ value to a solution value of 24 volts even though the initial square distribution was ± 15 volts and the maximum standard deviation of the gaussian distribution was $\sigma_\gamma = 10$ volts. This situation occurred often. The second display, illustrated in figure 11, provides information about every iteration. Here the values of the boundary cost function J_D , J_V , and J_P are shown for every iteration and the successful iterations. It can be noted that when

no improvement is obtained in the J vector, the search gradually enlarges until an improvement is found. At this point the search is localized. With this type of display it is easy to select a rational value for the lower limit of the standard deviation, σ_1 .

The convergence time to obtain a solution is certainly one of the central considerations in the random search method. Since this time is a random variable, data were taken to give suitable averages and some indication of their accuracy. In terms of the number of iterations, N , required for a solution, it was found from 70 solutions for the particular situation described above that

$$\begin{aligned}\bar{N} &= 7,724 \\ \sigma_N &= 5,142\end{aligned}$$

Thus, the average length of time to obtain a solution is about 1.25 minutes. The value of σ_N gives some indication of the accuracy to be expected for measurements of other situations. For example, if 25 trials are made for another situation, we would expect the standard deviation of the average to be $\sigma_{\bar{N}} = \sigma_N/\sqrt{25} \approx 1,000$. This relation is only an indication, however, since the standard deviation, σ_N , would not likely remain the same for other situations.

Next, we will examine some of the possible variations in the algorithm and its parameters to indicate their relative

effects on the convergence. Although these effects are studied for the given problem situation, the trends are generally valid for other situations.

First, the effect of the threshold strategy is shown in figure 12. The value of $\eta = 0$ corresponds to removing the threshold strategy. Also, a standard deviation is shown on the curve. It is worth noting that some improvement can be obtained by increasing η .

Fig. 12

Second, the effect of the one-step strategy, the end-search strategy, and an increased initial search size are given below with standard deviations as indicated. It is

	Average number of iterations
Algorithm with: one-step strategy	
end-search strategy	7,724 \pm 610
normal initial search size	
Algorithm without one-step strategy	12,707 \pm 1,150
Algorithm without end-search strategy	7,065 \pm 1,626
Algorithm with twice initial search size	9,971 \pm 766

seen that the one-step strategy is fairly effective, since its deletion increases convergence time by 65%. It was always beneficial in terms of convergence, costs so little in search time, and was, therefore, always employed. On the other hand, it is seen there that the end-search strategy is slightly

deleterious for the given situation and increases convergence time by 10%. This increase is not too significant, however, in view of the likely standard deviation. The end-search strategy appears to be of greatest value when the algorithm has iterated to the neighborhood of the minimum and has difficulty in meeting the boundary conditions; such a situation might exist perhaps in the neighborhood of a very sharp valley. In these cases, the end-search strategy greatly reduces the size of the search and enhances the certainty of finding the minimum. The effect of initial search size on convergence is interesting. It is surprising to note that an increase in the initial search size of 100% in all five adjoint variables (32 times larger volume) resulted in an increase of only 30% in convergence time.

The initial starting value J^0 is another parameter in the algorithm which is somewhat arbitrary. The effect of different starting values, in figure 13, shows that convergence is fairly flat within a wide range. At the lower values, the convergence time increases sharply because the search approaches the pure random search.

Results for Orbit-Transfer Problem

In this section we will illustrate in some detail the type of results obtained for one particular situation, the same situation as in the preceding section.

A typical solution which was obtained by the random search algorithm is illustrated in figure 14 by the rather complete set of time histories; shown are the system states in the rotating and inertial coordinate systems, the control vector, and the adjoint variables. Perhaps the most interesting are the velocities x_2 and x_4 in the rotating frame which can be seen to be reduced from large values of around -900 m/s and 400 m/s to small values in order to reduce the inertial velocity components, V_R and V_T , to near their desired values. Also the quantity $r - r_c$, the difference between the vehicle terminal radius and the desired circular orbit radius can be seen to have been reduced to a small value so that the vehicle will end up close to the desired orbit. The terminal inertial value y is slightly positive indicating the desired orbit was attained slightly past pericynthian. The optimal control vector is also of interest; it is indicated that the thrust should be turned on at about 260 seconds before the fixed final time and directed as shown.

< Fig. 14

Average performance and its variation are perhaps more significant than an individual solution. Observation of 70 solutions revealed that their time histories were only slightly different. In terms of the end states and fuel used, the following statistics were obtained:

Ave fuel	=	10,230	±80 kg
Ave x	=	1,935	±5 km
Ave y	=	33	±8 km
Ave $r(T) - r_c$	=	6	±5 km
Ave $V_T - r_c\omega$	=	2	±7 m/s
Ave V_R	=	1	±12 m/s

It is worth noting here (from x , y , and $r - r_c$) that the set of reachable points is centered slightly further out than the desired circular orbit and slightly past pericynthian; the standard deviation shows this set is confined to a very small region. The inertial velocities indicate that they have been substantially reduced from the initial total magnitude of approximately 2400 m/s. The fuel data are interesting because the figures are so close to the idealized minimum-fuel impulsive orbit transfer of 10,120 kg.

It is significant to note that from this same set of 70 solutions, the solutions for the initial condition adjoint vector $p(0)$ were vastly different even though the time histories of the system states were so close. The extremes observed as well as the $p(0)$ vector corresponding to figure 14 are as follows.

Adjoint variable	Extremes	Values for figure 14
p_1	-31.98 to +7.15	-11.36
p_2	+1.08 to +14.67	+8.96
p_3	-8.33 to +35.60	-.54
p_4	-11.94 to +5.45	-9.25
p_5	-7.98 to +8.11	-6.30

These results are significant because they illuminate the nature of the mapping $p(0) \rightarrow J$. In geometrical terms, discussed in more detail later, these results mean that the boundary cost function hypersurfaces have many stalactites protruding below the ϵ level.

Variations in the time allowed to reach the desired orbit were studied with much the same type of results as in the preceding two paragraphs. Solutions were obtained for times from 550 seconds to 850 seconds (the time to reach pericynthian with zero control was 534 seconds). The main difference in results obtained was that as the allowable time increases, the set of reachable points as given by X and Y moves further around the desired circular orbit and the fuel requirements appear to increase. For example, for $T = 850$ seconds, $X \approx 1828$ km, $Y \approx 658$ km, and the fuel required increased by about 300 kg. When the allowable time is outside the range given above, no solutions could be obtained.

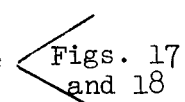
Results for Other Situations

The algorithm was experimentally tested on a variety of situations. They include different initial points along each of the three trajectories as indicated by the grid of points in figure 9, different allowable times for the vehicle to reach the desired point, and variations in some of the vehicle design parameters. A sampling of some of these data and a few comments are appropriate.

Figures 15 and 16 show two different solutions to the same problem in which the starting point is further along on

Figs. 15
and 16

orbit 2 (-13°) than in the previous two sections. These solutions are interesting because of the unusual control functions found. One of the control solutions has two thrusting periods (on-off-on) with the initial thrusting being quite long; the other solution has only one thrusting period (on-off) and again the initial thrusting is long. In contrast, most of the solutions found for other situations have only a single thrusting period of the off-on type.

Next, figures 17 and 18 illustrate solutions which were  Figs. 17 and 18 obtained on orbits 1 and 3. On orbit 1, the solutions showed there was always one thrusting period of the off-on type (see fig. 17) for all starting points along the orbit and all times T for which solutions could be obtained; the fuels used were close to the values obtained on orbit 2. On orbit 3, the solutions nearly always consisted of the more unusual two thrusting periods of the on-off-on type (see fig. 18). It was also noted that solutions were quite difficult to obtain on orbit 3 and that the fuels required were approximately 1500 kg greater than on orbits 1 and 2. No explanation was apparent.

Finally, it was demonstrated that the random search approach could be of considerable value in preliminary vehicle design. For this purpose the effects of varying some of the initial vehicle design parameters were investigated, namely, thrust level and mass. It is significant to note that solutions

could readily be obtained with engines of 75% and 50% of the normal thrust capability used in the previous portions of the study. Similarly, there were no difficulties in obtaining solutions for different initial vehicle masses.

Boundary Cost Function Surfaces

Perhaps the most illuminating results were revealed by an experimental study of the mapping: $p(0) \rightarrow J$. Geometrically this mapping can be interpreted as three hypersurfaces (representing the components of the metric J), which are functions of the five adjoint initial conditions. The character of the cross sections through these surfaces at a solution point is given in figures 19 and 20 for the two different problem situations indicated. These curves were obtained by slowly varying one of the adjoint variables from -100 volts to +100 volts while all the others were fixed at their respective solution values. In this manner cross sections through the hypersurfaces passing through a solution point were plotted. Note the dip in all three surfaces at the optimum value of the adjoint variable. The interest in these curves is, of course, in the rather irregular, multi-valleyed nature of the surfaces as well as the rather narrow valleys surrounding the optimum point. The sharpness or narrowness of this valley gives an indication of the difficulty in finding the solution. Further, these surfaces clearly indicate the difficulties which gradient methods would have because of the likely possibility of "hanging up" in the wrong valley. The two-dimensional cross sections

Figs. 19
and 20

shown give only a hint of the difficulties to be encountered in the actual six-dimensional space.

Another more pictorial representation (for the same problem situation as in figure 19) is given by the three-dimensional views in figure 21. Here are shown the boundary cost function surfaces in the p_1, p_2 plane and in the p_1, p_4 plane while all other adjoint variables are fixed at their solution values. These results illustrate the character of the surfaces away from the optimum point in two directions instead of only one direction.

Fig. 21

Further clues as to the nature of the boundary cost function surfaces are revealed by previously presented data in which it was shown that many distinctly different $p(0)$ vectors result in solutions. The significance of this type data is that it shows that the surfaces have many stalactites which protrude below the ϵ level.

CONCLUDING REMARKS

It is worth emphasizing that the objective in the present study has been to devise and demonstrate a feasible method of implementing the Maximum Principle based on an adaptive random search algorithm. It has been demonstrated that the computational requirements are most effectively implemented on a hybrid computer system. The method has been shown to be a practical approach to generating explicit optimal solutions for a moderately complex example problem under a wide range of situations. The

versatility and ease in obtaining solutions makes it a valuable approach for preliminary system design because it allows one to study the tradeoffs among various initial design choices in terms of performance. More generally, the method could be applied to any two-point boundary value problem or to parameter optimization of systems with given configurations.

More effective utilization of the method either in an on-line or off-line implementation could be realized by an increase in the iteration rate. Such increases will in general have to be the result of advances in hybrid systems. However, a moderate increase in rate could be realized with the present system by careful redesign of the program.

Undoubtedly the most important limitation of the method is due to the minimum resolution of the analog computer. Thus certain problems with excessive dynamic range might be excluded.

The approach has a number of advantages over alternate approaches. It is conceptually simple, straightforward, and easily implemented. It can be applied uniformly within its limitations. No restrictions are placed on the boundary surface, and an analytical expression for the surface is not required. The algorithm's local and global search properties provide it with the ability to jump from one local valley to another so that "hanging up" in local valleys is avoided.

REFERENCES

1. Knapp, C. H. and Frost, P. A.: Determination of Optimal Control and Trajectories Using the Maximum Principle in Association With a Gradient Procedure. Joint Automatic Control Conference, Stanford University, June 24-26, 1964.
2. Brooks, Samuel H.: A Discussion of Random Methods for Seeking Maxima. Operations Research, vol. 6, no. 2, March 1958.
3. Bekey, G. A.: Parameter Optimization by Random Search Using Hybrid Computer Techniques. AFIPS Conference Proceedings, vol. 29, 1966.
4. Maybach, R. L.: Solution of Optimal Control Problems on a High-Speed Hybrid Computer. Simulation, vol. 7, no. 5, November 1966.
5. Pontryagin, L. S.; Boltyanskii, V. G.; Gamkrelidze, R. V; and Mischenko, E. F.: The Mathematical Theory of Optimal Processes. Interscience Publishers, 1962.
6. Athans, M. and Falb, P. L.: Optimal Control. McGraw-Hill Book Company, 1966.
7. Rozonoer, L. I.: L. S. Pontryagin's Maximum Principle in the Theory of Optimal Systems I, II, III. Automation and Remote Control, 20, October, November, December 1959.
8. Zadeh, L. A.: Optimality and Non-Scalar-Valued Performance Criteria. IEEE Transactions on Automatic Control, vol. AC-8, number 1, January 1963.

FIGURE LEGENDS

Figure 1.- Hybrid system block diagram for random search method.

Figure 2.- Typical boundary cost function surface.

Figure 3.- Hybrid system hardware.

Figure 4.- Sequencing of events during one iteration.

Figure 5.- Algorithm flow graph.

Figure 6.- Reset loop.

Figure 7.- Search loop.

Figure 8.- End-state loop.

Figure 9.- Geometry of orbit-transfer problem.

Figure 10.- Behavior of successful iterations during search.

Figure 11.- Behavior of every iteration during search.

Figure 12.- Effect on convergence of threshold strategy.

Figure 13.- Effect on convergence of varying initial value J^0 .

Figure 14.- Time history solutions; orbit 2, $x(0)$ at -37° ,

$T = 600$ sec.

Figure 15.- Time history solutions 1; orbit 2, $x(0)$ at -13° ,

$T = 400$ sec.

Figure 16.- Time history solutions 2; orbit 2, $x(0)$ at -13° ,

$T = 400$ sec.

Figure 17.- Time history solutions; orbit 1, $x(0)$ at -60° ,

$T = 971$ sec.

Figure 18.- Time history solutions; orbit 3, $x(0)$ at -60° ,

$T = 1075$ sec.

Figure 19.- Two-dimensional cross sections of boundary hyper-surface; orbit 2, $x(0)$ at -37° , $T = 600$ sec.

Figure 20.- Two-dimensional cross sections of boundary hyper-surface; orbit 3, $x(0)$ at -60° , $T = 1075$ sec.

Figure 21.- Three-dimensional cross sections of boundary hyper-surface; orbit 2, $x(0)$ at -37° , $T = 600$ sec.

(a) p_1, p_2 plane.

Figure 21.- Concluded.

(b) p_1, p_4 plane.

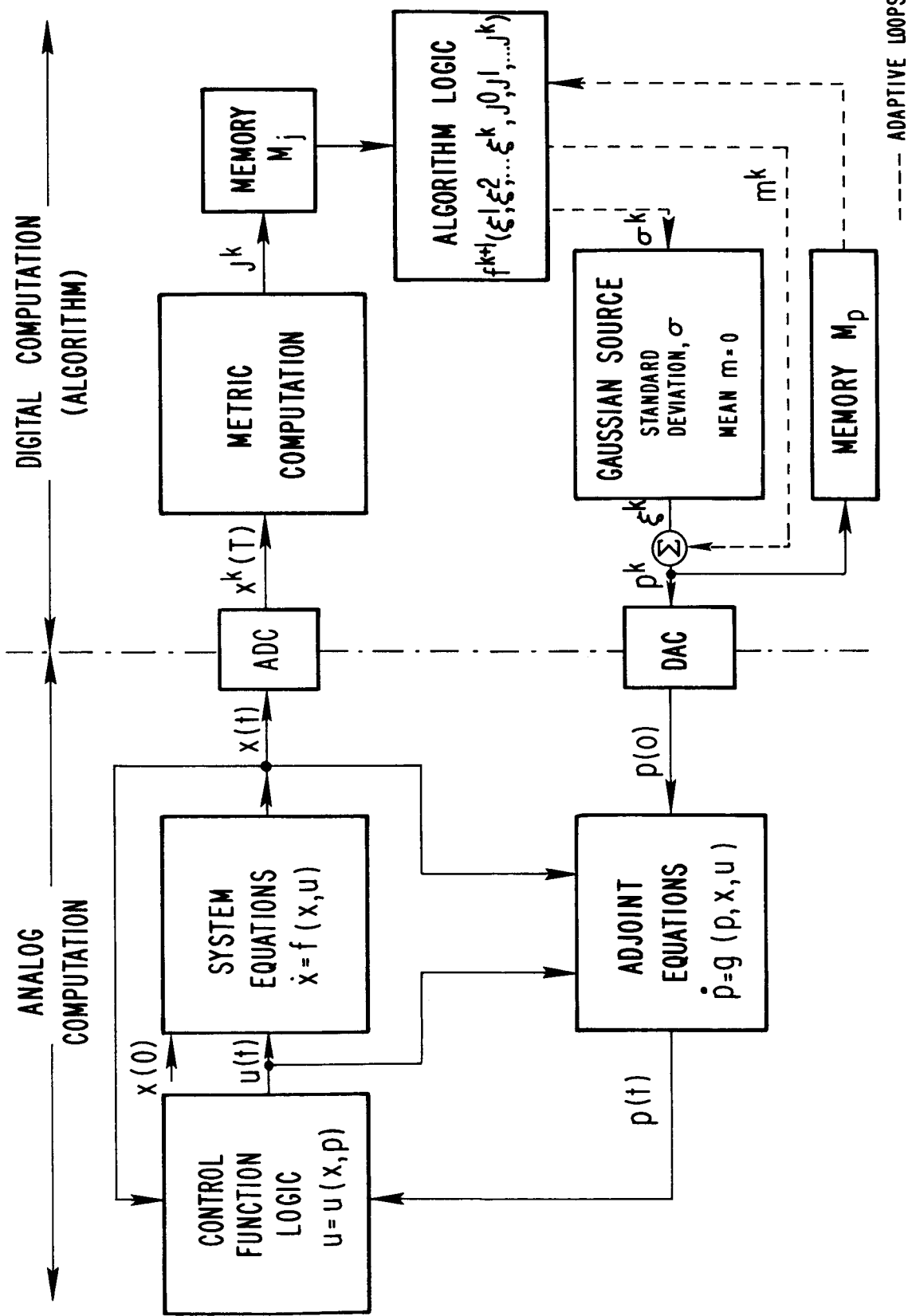


Figure 1.- Hybrid system block diagram for random search method.

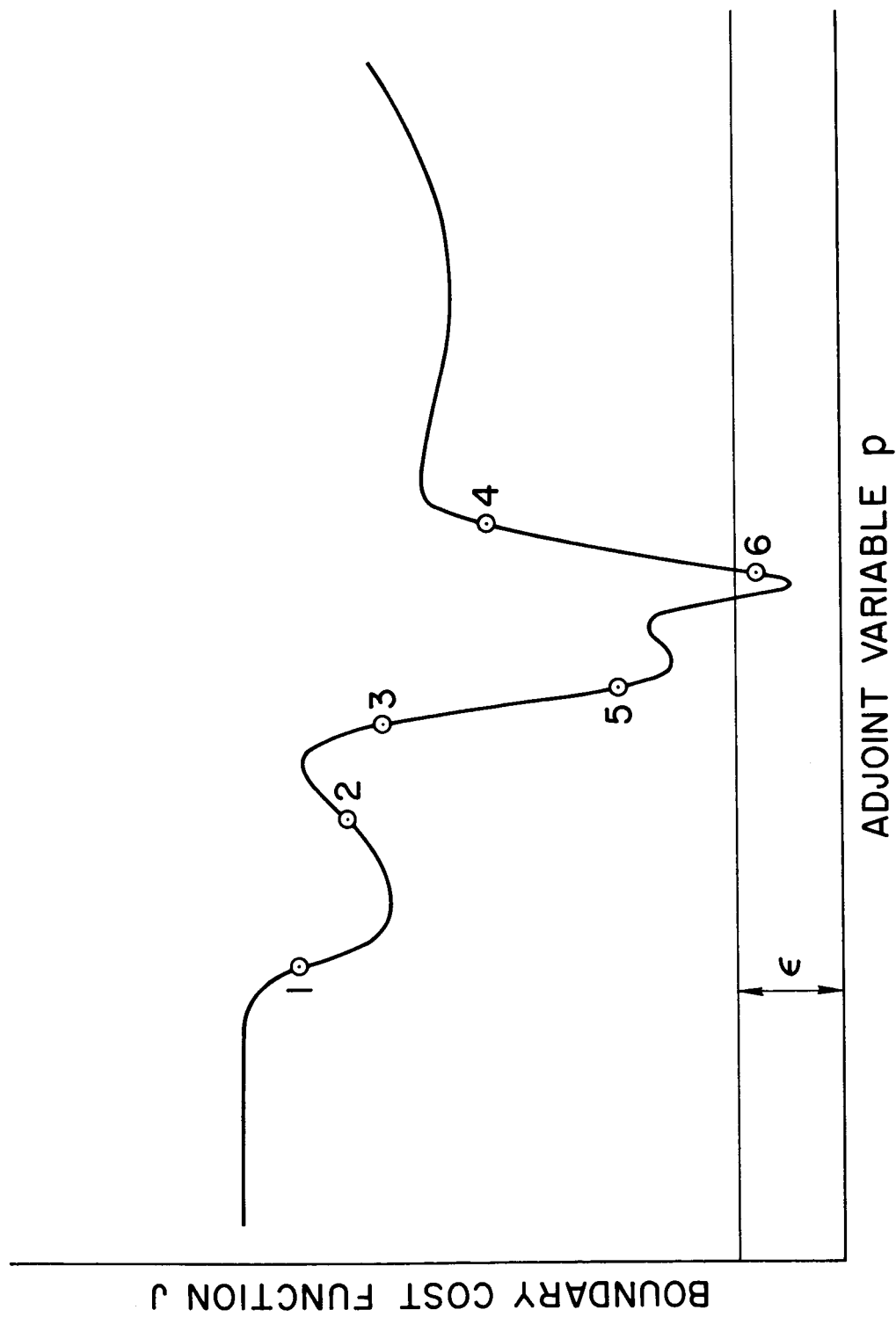


Figure 2.- Typical boundary cost function surface.

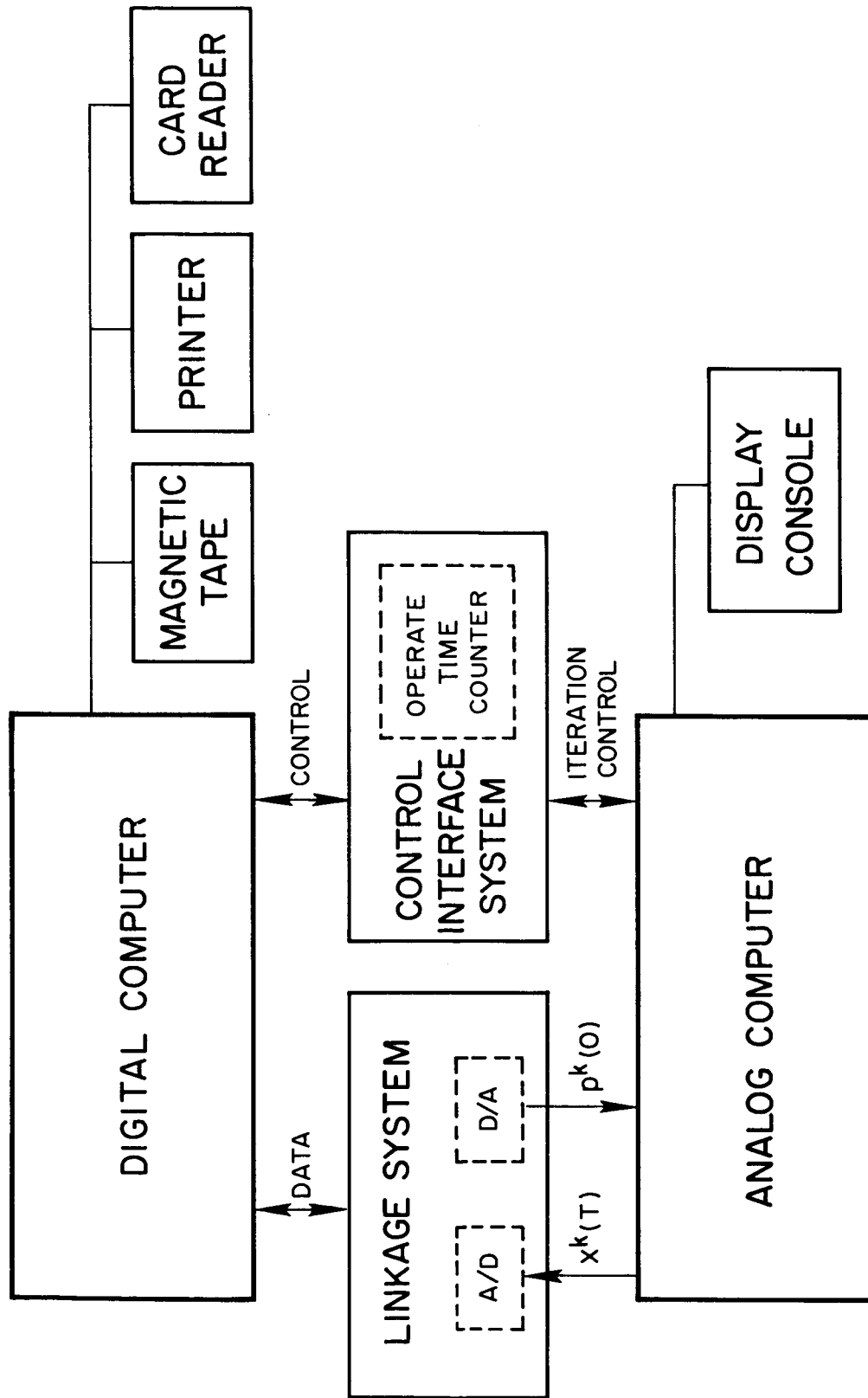


Figure 3.-- Hybrid system hardware.

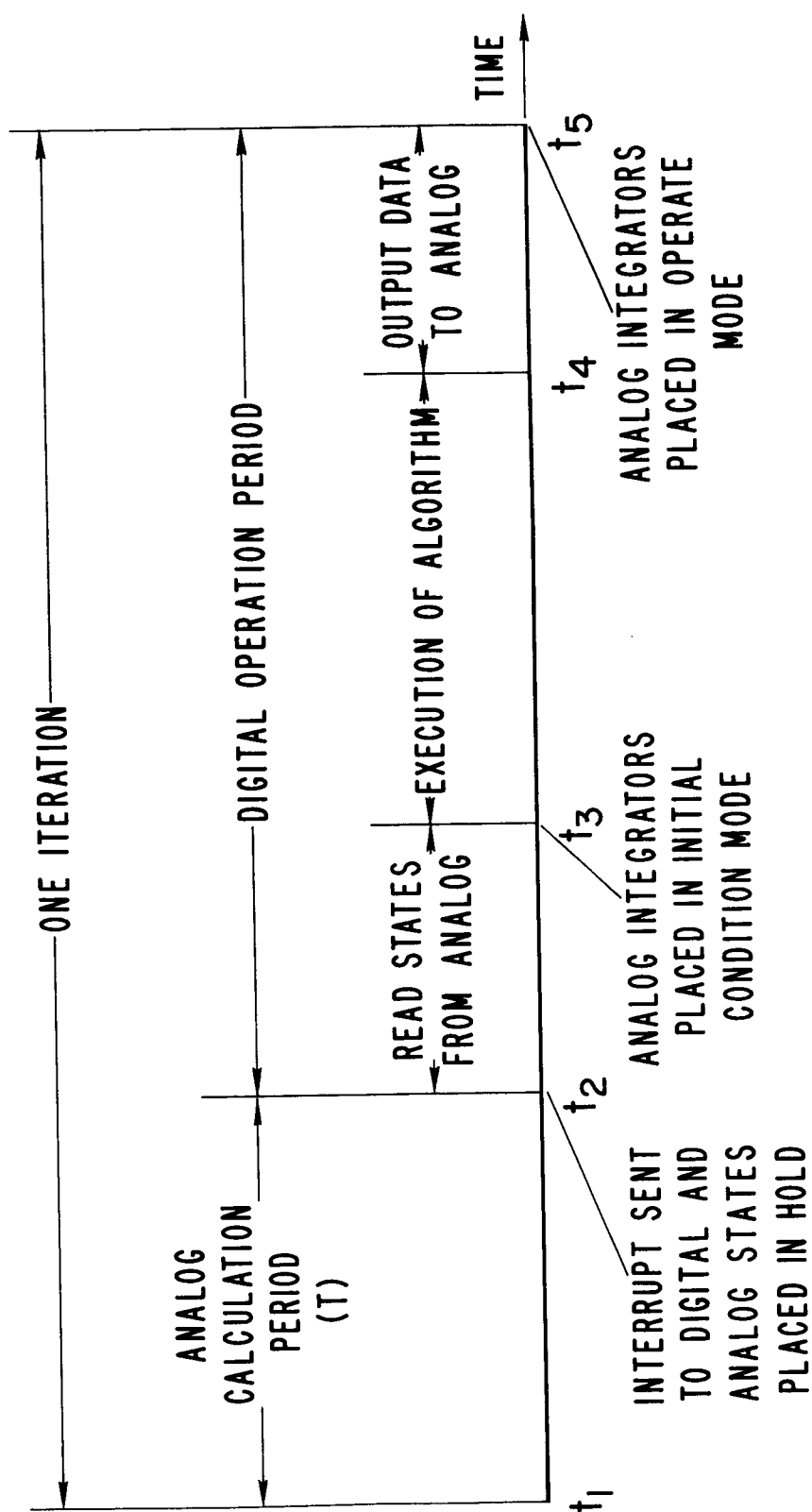


Figure 4.- Sequencing of events during one iteration.

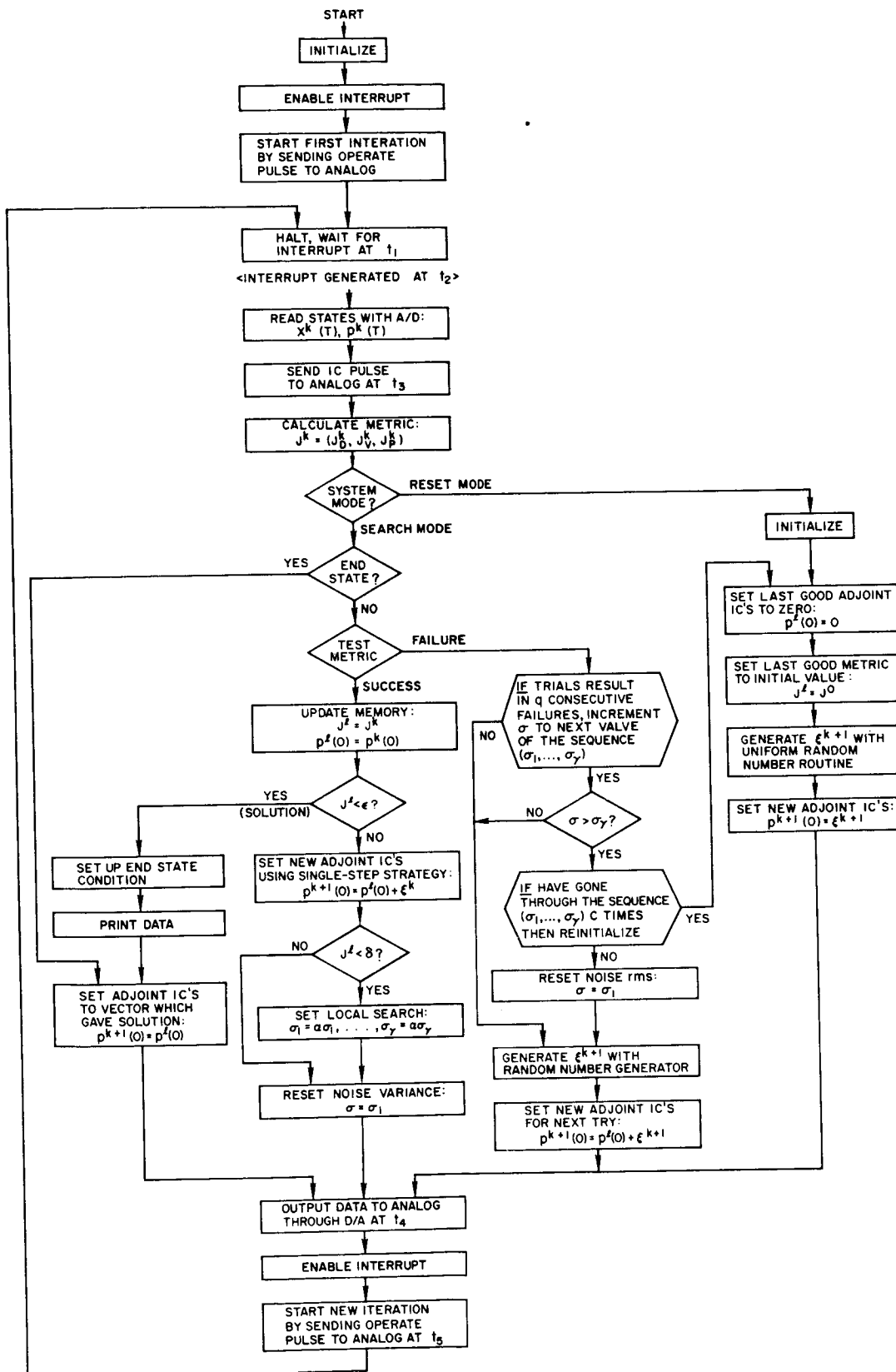


Figure 5.- Algorithm flow graph.

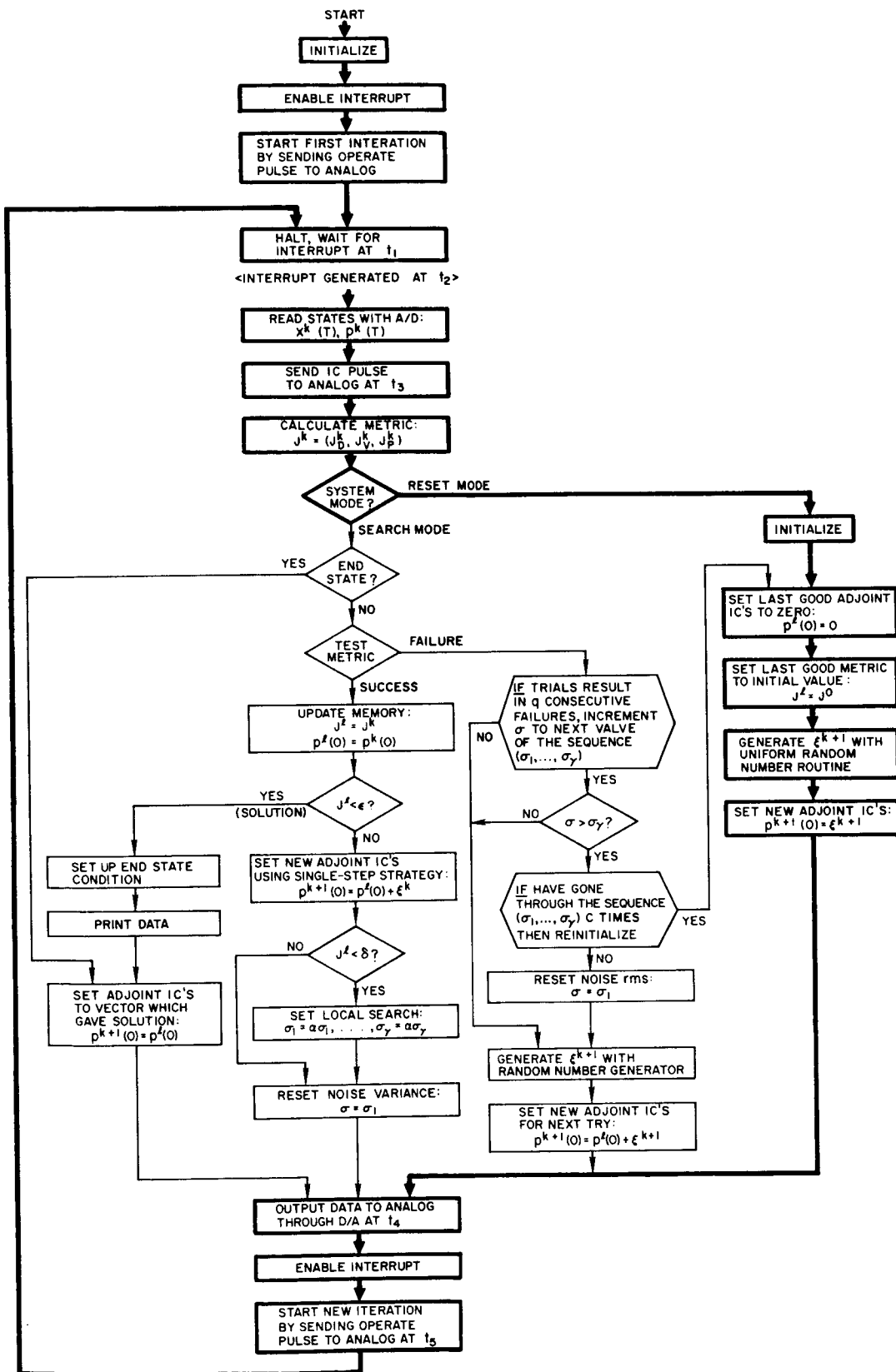


Figure 6.- Reset loop.

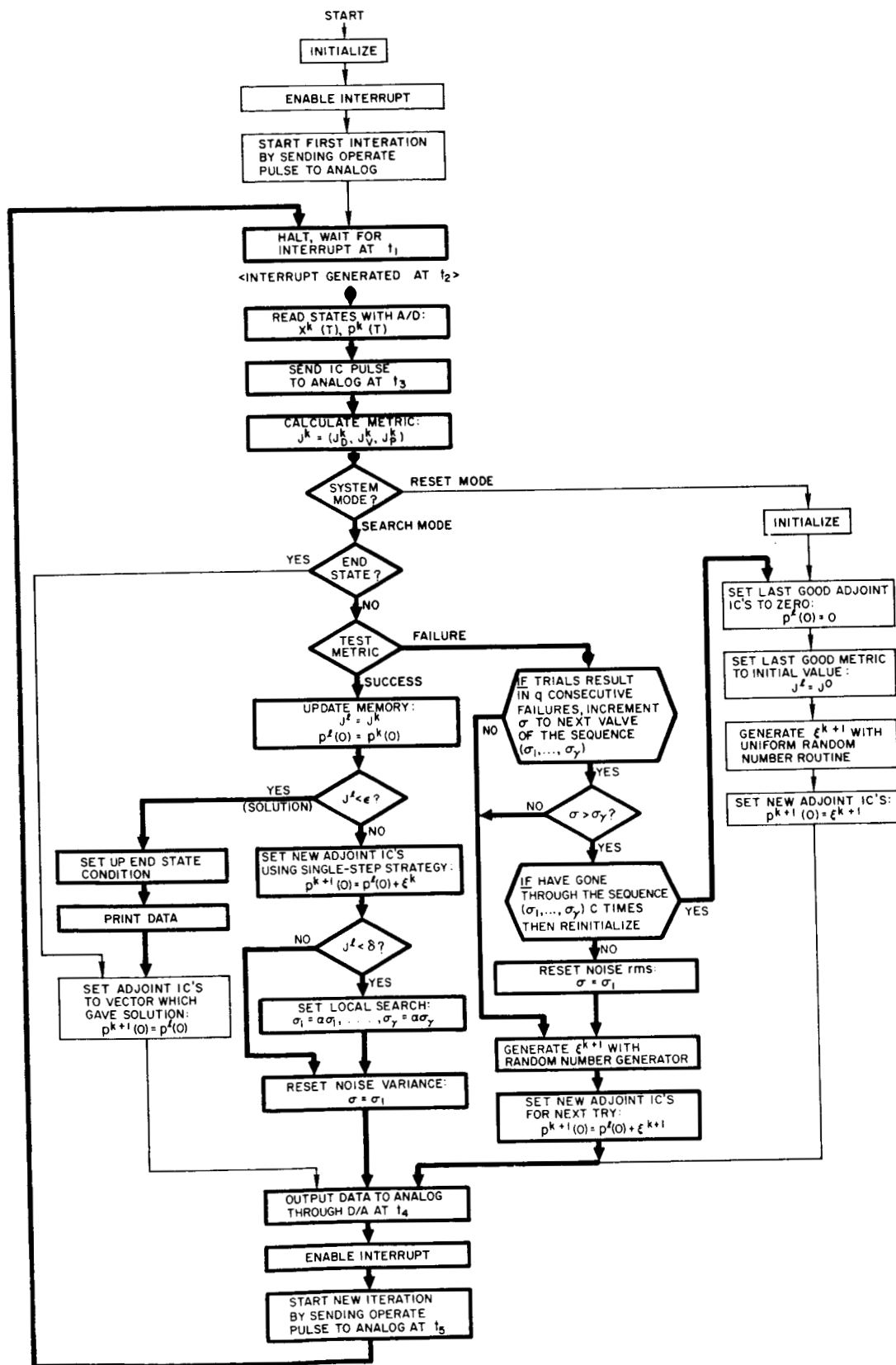


Figure 7.- Search loop.

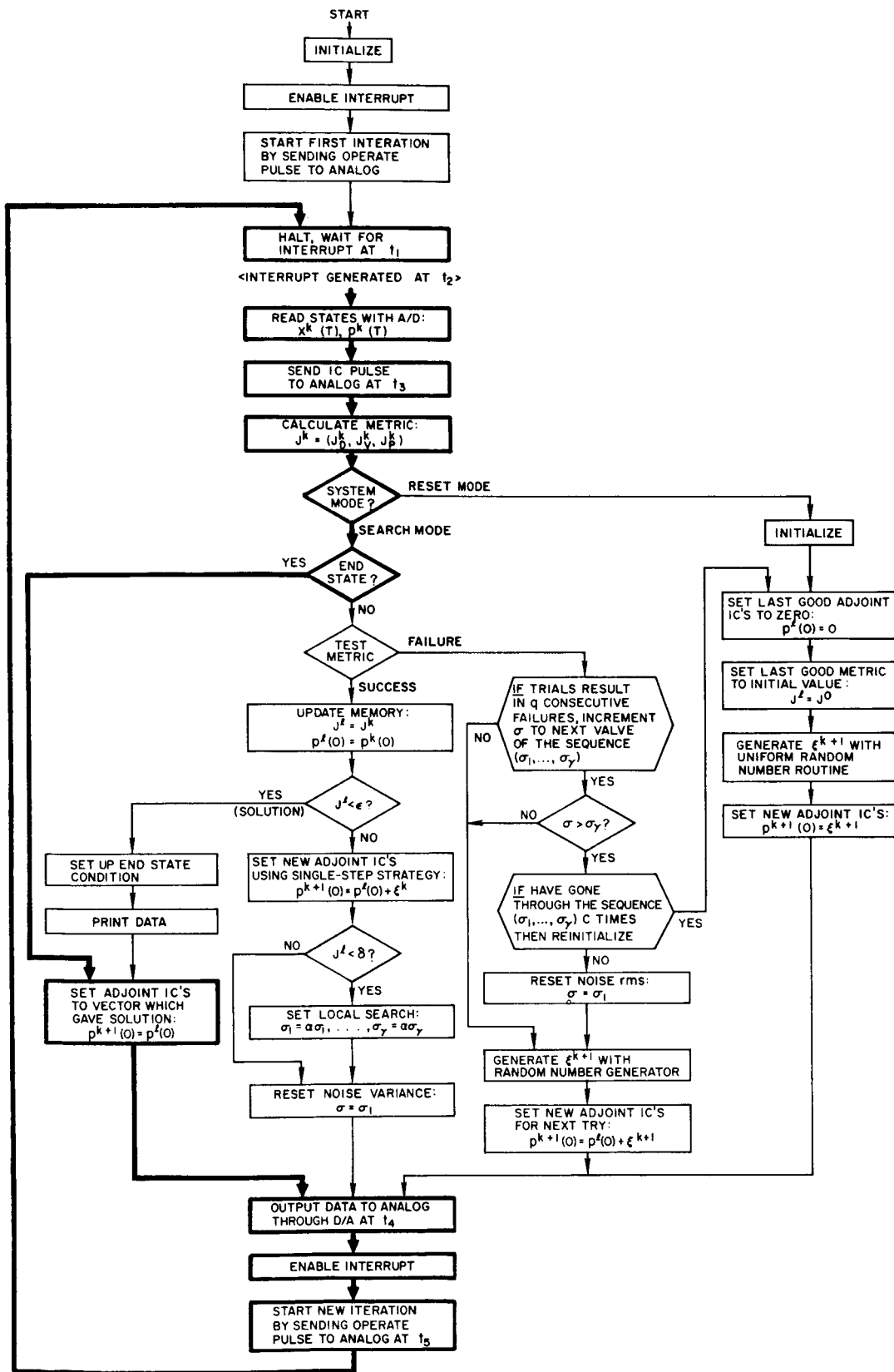


Figure 8.- End-state loop.

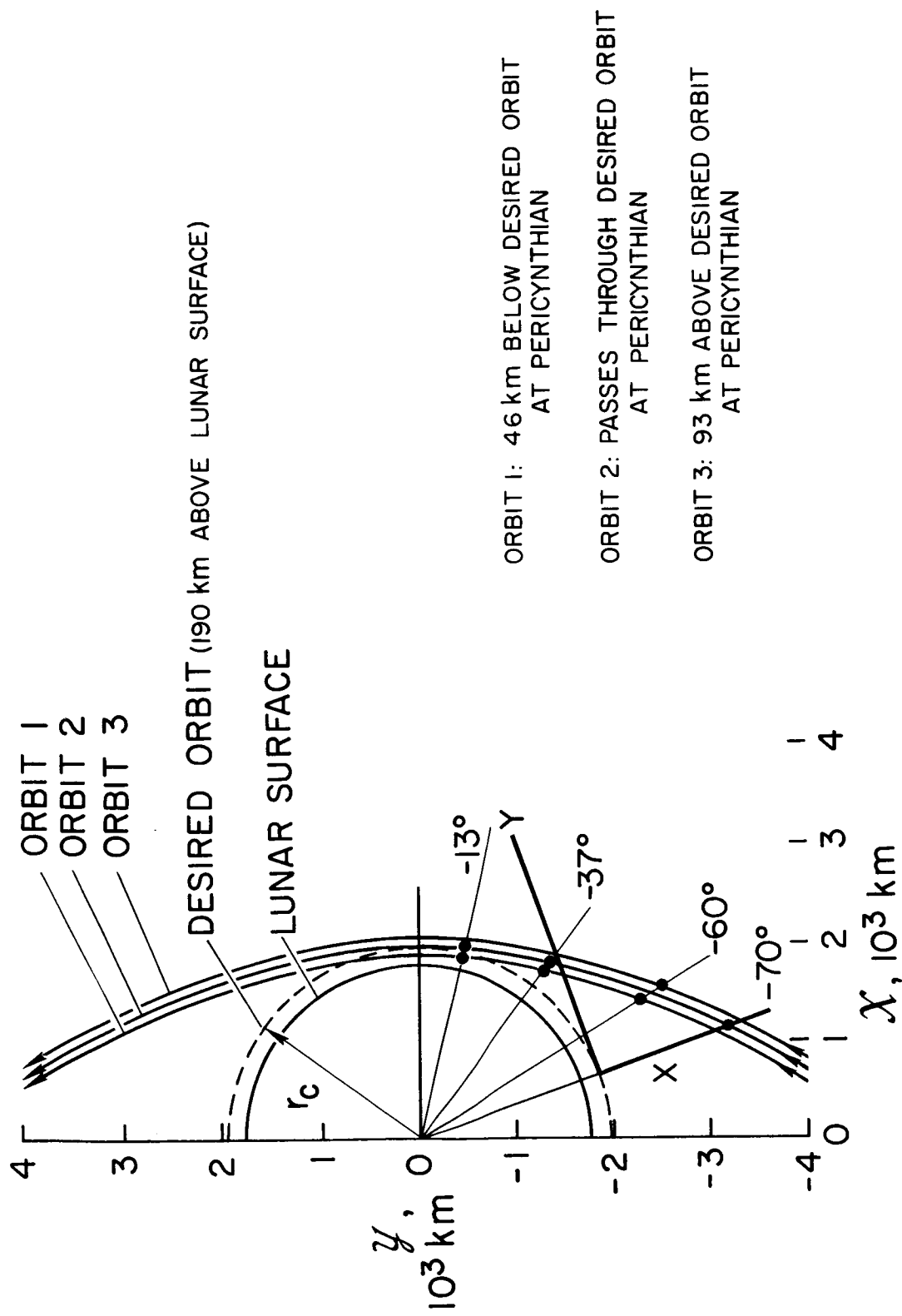
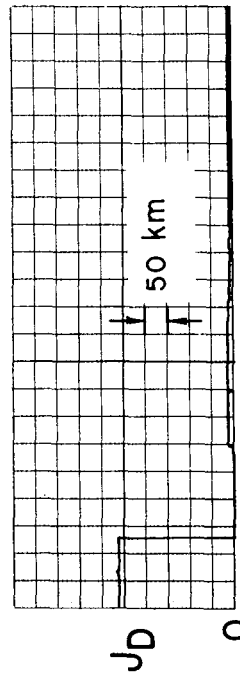
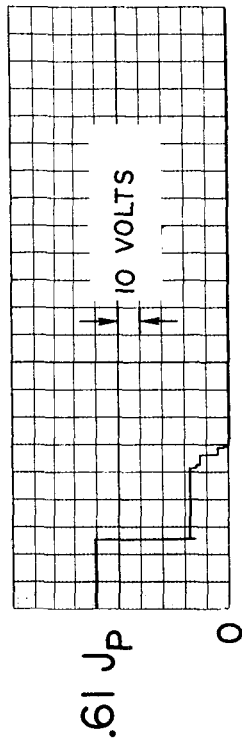
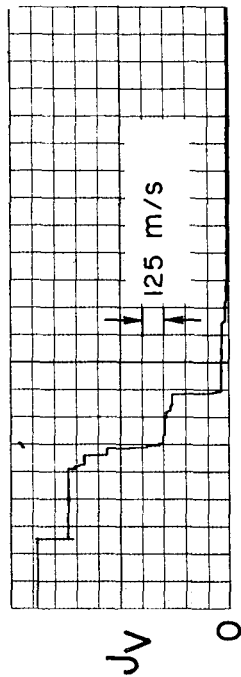
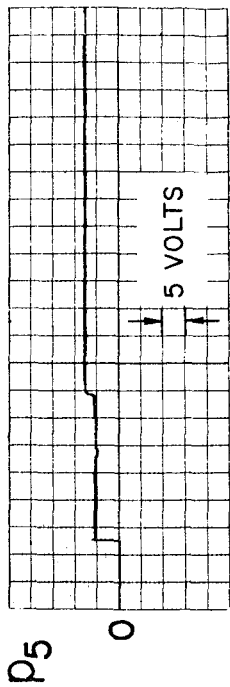
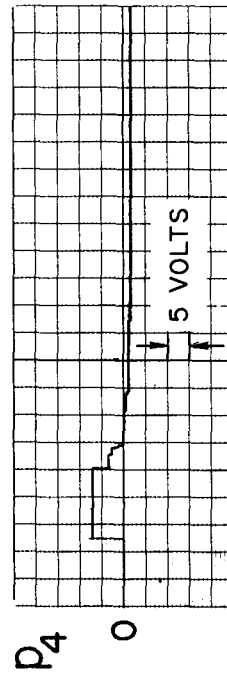
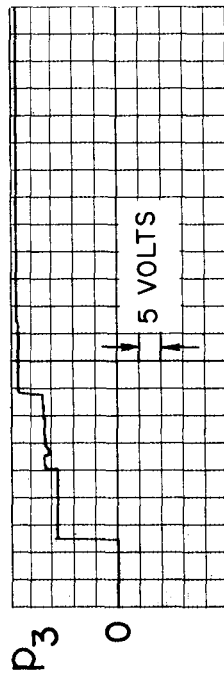
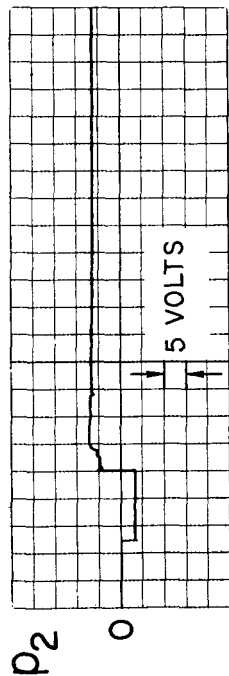
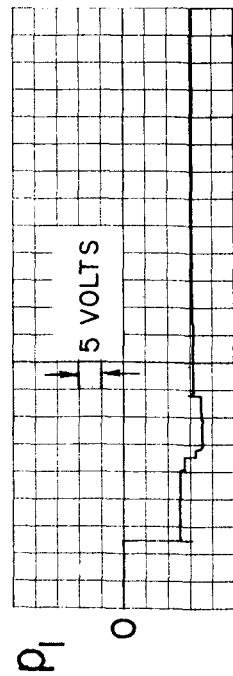


Figure 9.- Geometry of orbit-transfer problem.



TIME, 1 DIV = 5 sec

Figure 10.- Behavior of successful iterations during search.

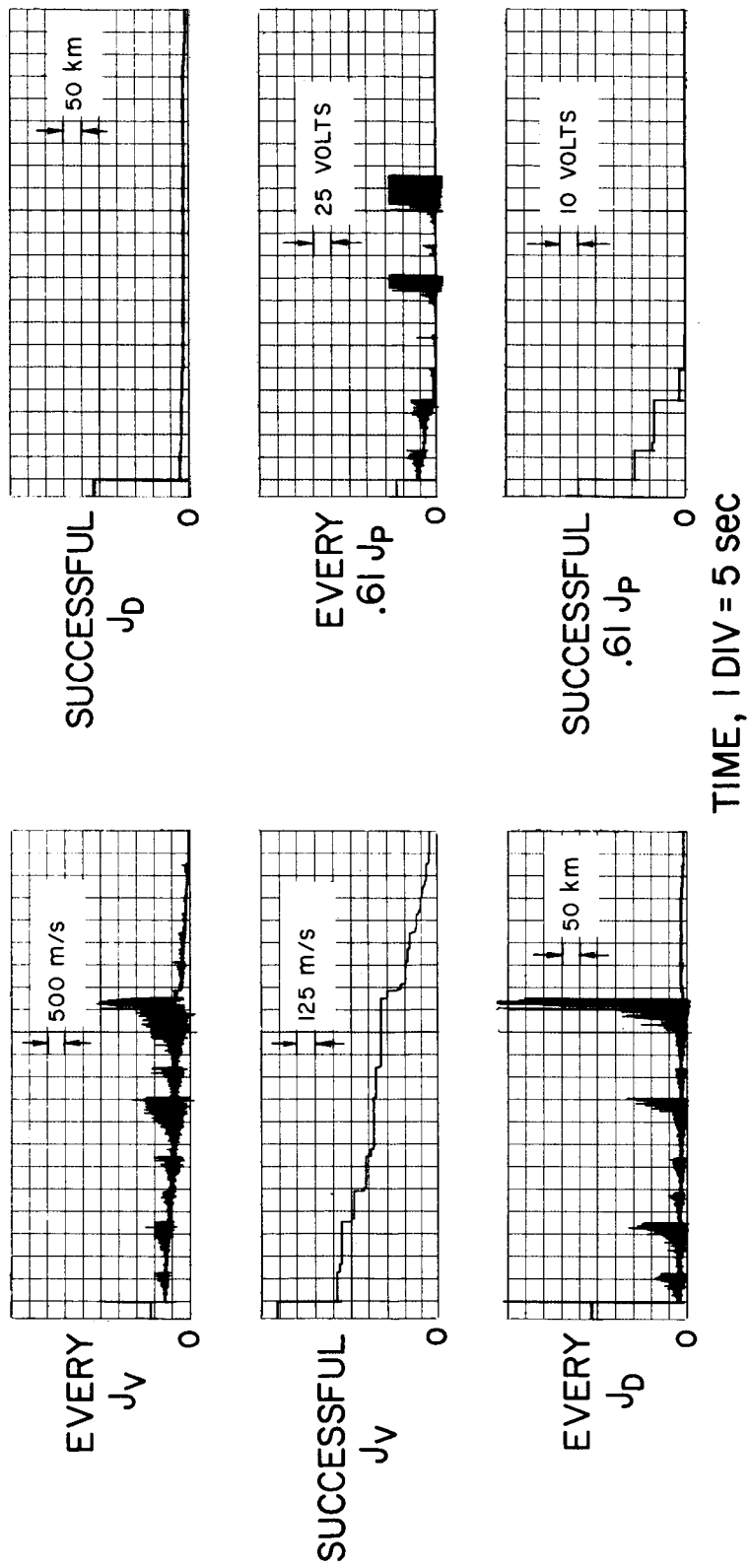


Figure 11.- Behavior of every iteration during search.

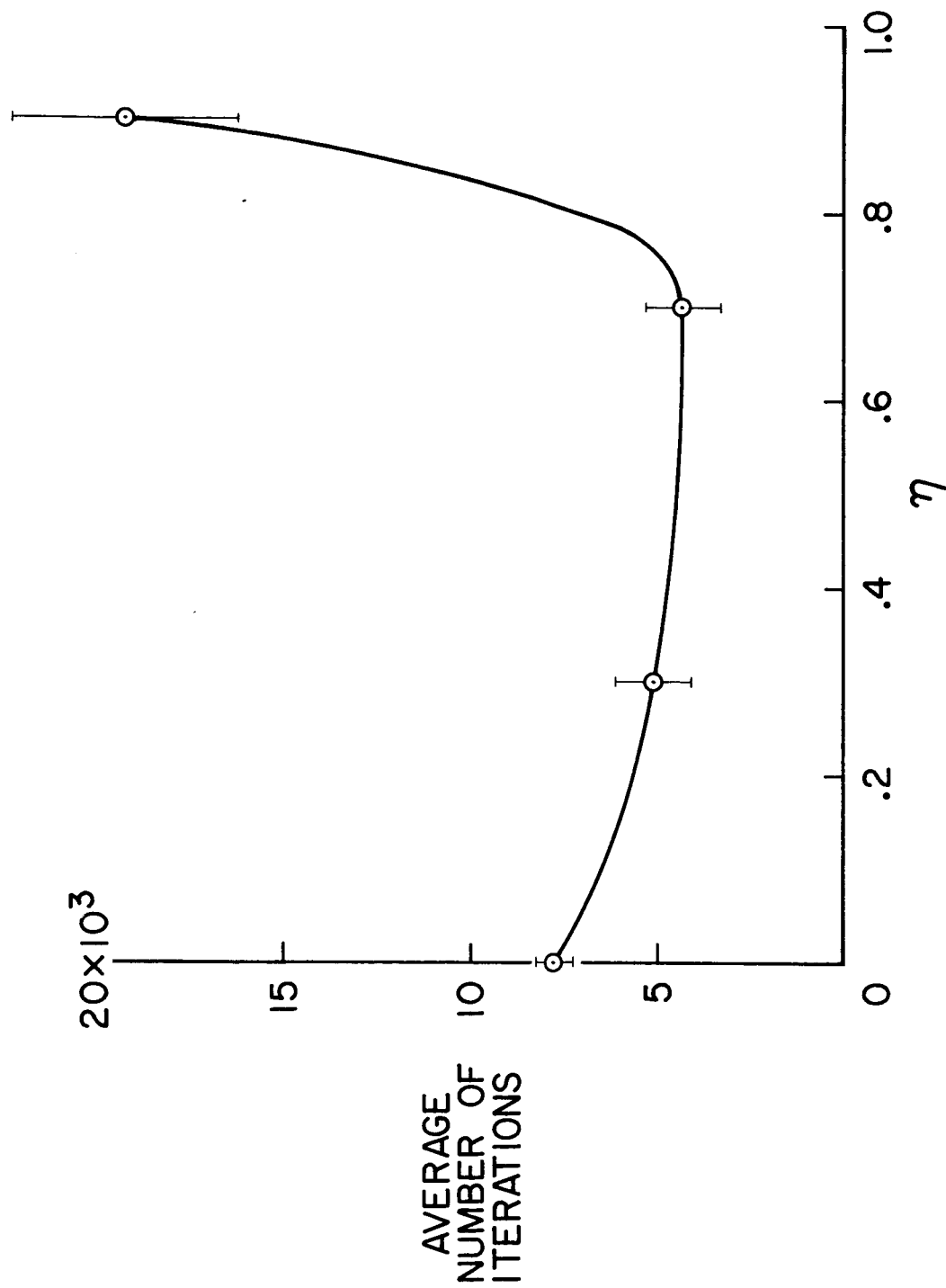


Figure 12.- Effect on convergence of threshold strategy.

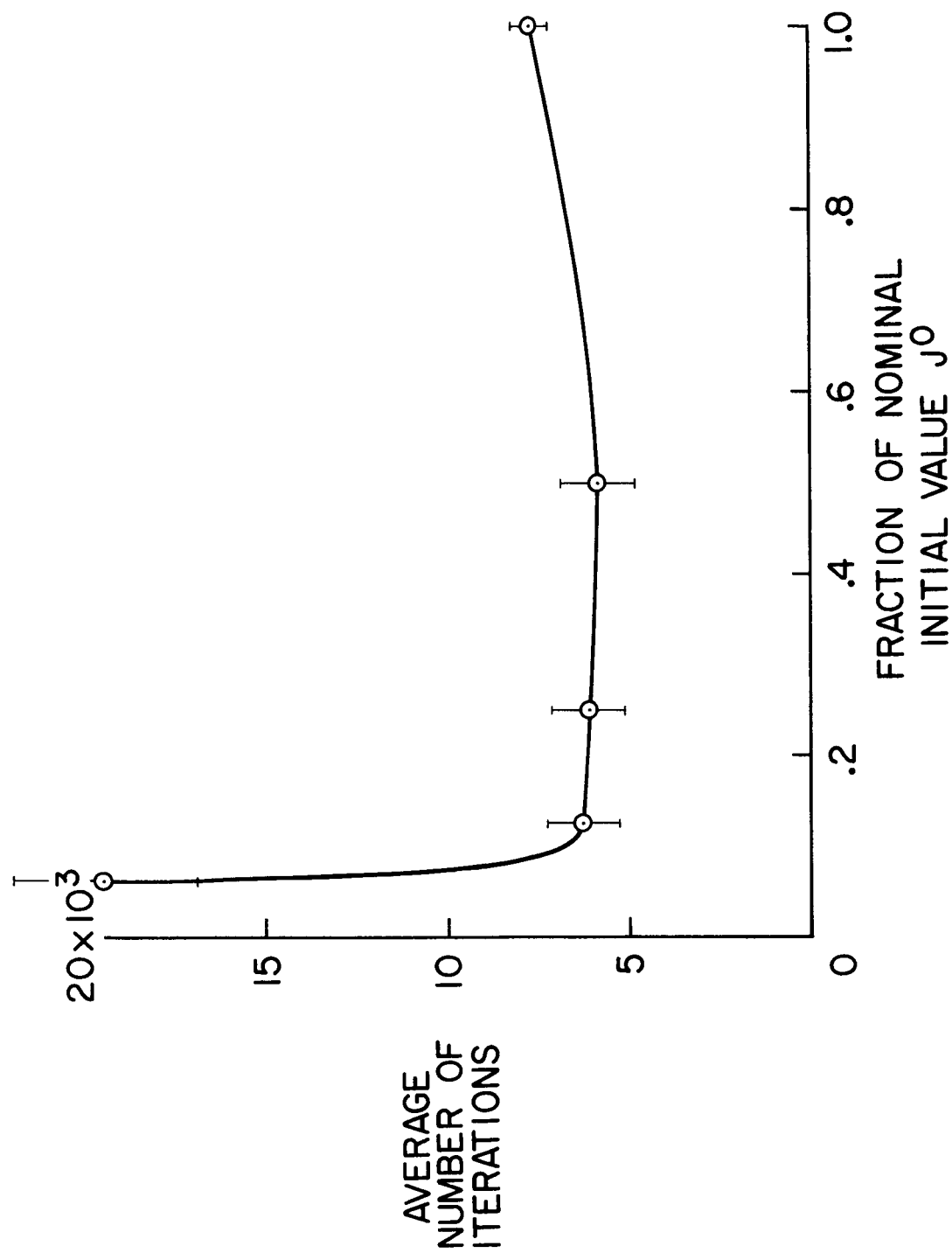
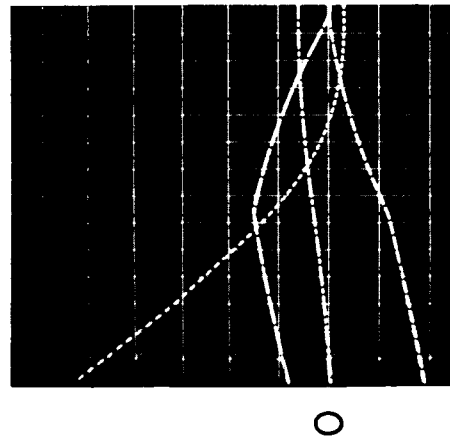


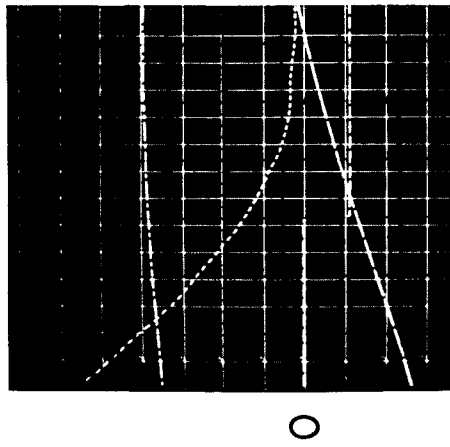
Figure 13.- Effect on convergence of varying initial value J^0 .

---- x_1 | DIV = 50 km
 --- x_2 | DIV = 500 m/s
 --- x_3 | DIV = 500 km
 — x_4 | DIV = 500 m/s

---- $r-r_c$ | DIV = 50 km
 --- $-u_3$ | DIV = 25 kg/s
 --- χ | DIV = 500 km
 — γ | DIV = 500 km



(a) STATES IN ROTATING
 COORDINATE SYSTEM

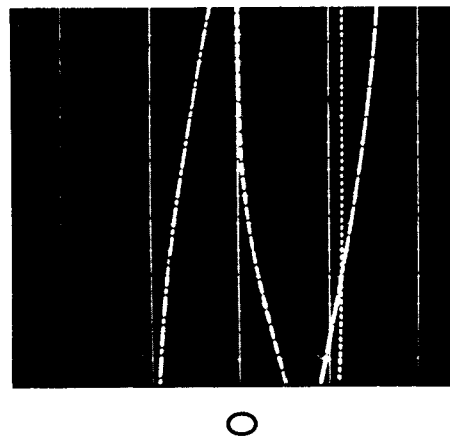


(b) INERTIAL COORDINATES

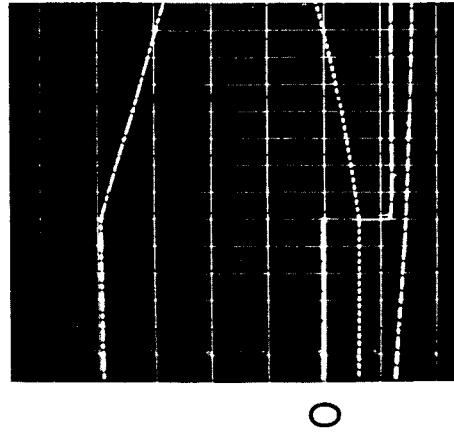
Figure 14.- Time history solutions; orbit 2, $x(0)$ at -37° , $T = 600$ sec.

----- p_1 1 DIV = 10 volts
 ---- p_2 1 DIV = 10 volts
 ---- p_3 1 DIV = 1 volt
 ---- p_4 1 DIV = 10 volts

----- p_5 1 DIV = 10 volts
 ---- $-||P||$ 1 DIV = 10 volts
 ---- x_5 1 DIV = 10,000 kg
 ---- $-u_3$ 1 DIV = 25 kg/s



0 T
(c) ADJOINT STATES



0 T
(d) MISCELLANEOUS STATES

Figure 14.- Continued.

----- SWITCHING FUNCTION, V

---- u_1 | DIV = .2

---- u_2 | DIV = .2

—— $-u_3$ | DIV = 25 kg/s

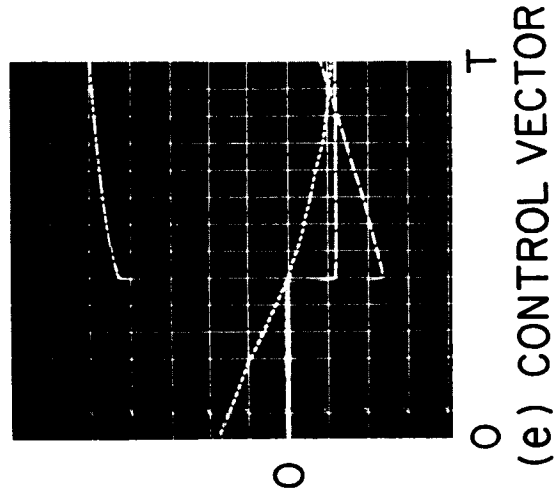
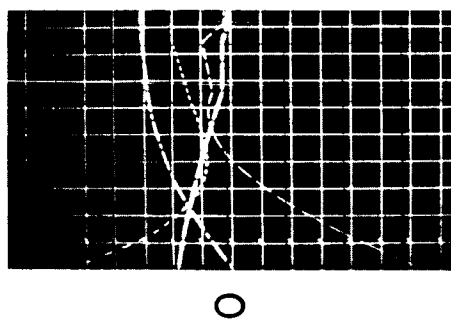
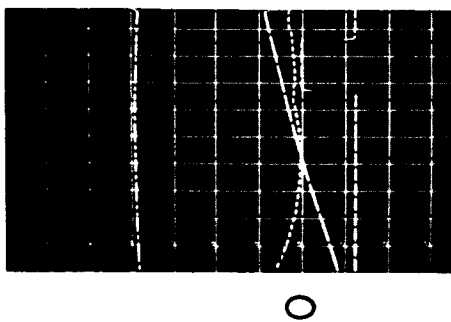


Figure 14.- Concluded.

----- x_1	DIV = 5 km	----- $r-r_c$	DIV = 50 km
---- x_2	DIV = 50 m/s	---- $-u_3$	DIV = 25 kg/s
- - - x_3	DIV = 50 km	- - - x	DIV = 500 km
— x_4	DIV = 500 m/s	— y	DIV = 500 km



(a) STATES IN ROTATING
COORDINATE SYSTEM

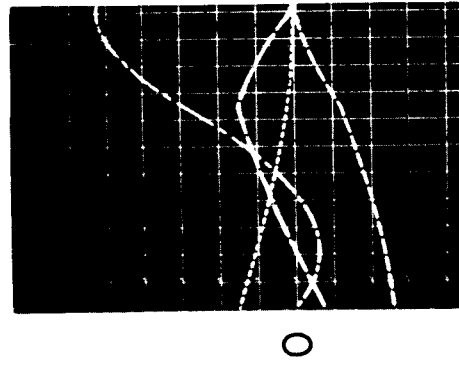


(b) INERTIAL
COORDINATES

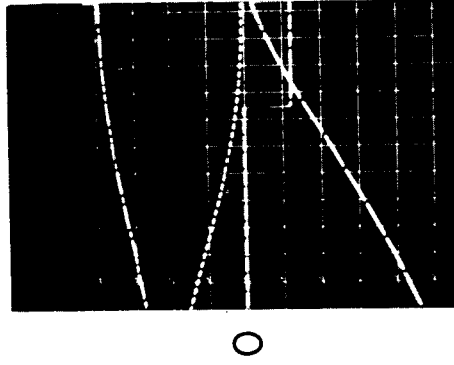
Figure 15.- Time history solutions 1; orbit 2, $x(0)$ at -13° , $T = 400$ sec.

---- x_1 | DIV = 500 km
 ---- x_2 | DIV = 500 m/s
 ---- x_3 | DIV = 50 km
 — x_4 | DIV = 500 m/s

---- $r-r_c$ | DIV = 500 km
 ---- $-u_3$ | DIV = 25 kg/s
 ---- x | DIV = 500 km
 — y | DIV = 500 km



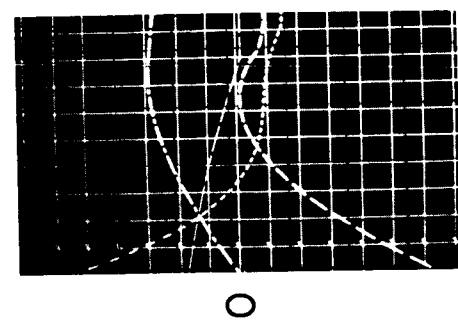
(a) STATES IN ROTATING COORDINATE SYSTEM



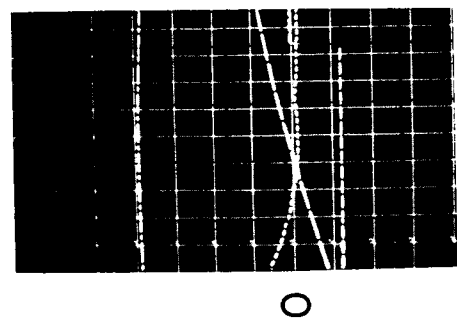
(b) INERTIAL COORDINATES

Figure 17.- Time history solutions; orbit 1, $x(0)$ at -60° , $T = 971$ sec.

----- x_1	DIV = 5 km	----- $r-r_c$	DIV = 50 km
----- x_2	DIV = 50 m/s	----- $-u_3$	DIV = 25 kg/s
----- x_3	DIV = 50 km	----- x	DIV = 500 km
----- x_4	DIV = 500 m/s	----- y	DIV = 500 km



(a) STATES IN ROTATING
COORDINATE SYSTEM

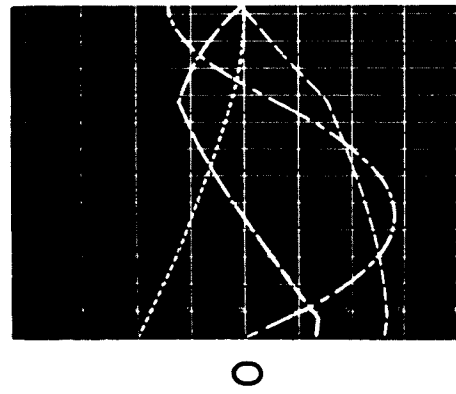


(b) INERTIAL
COORDINATES

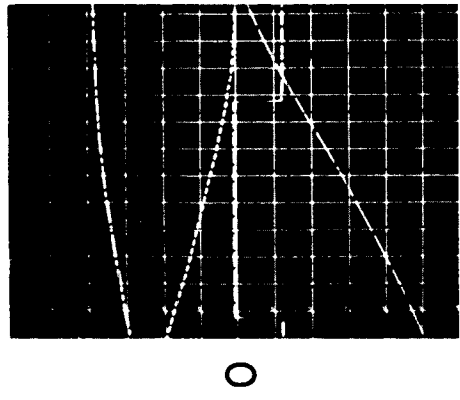
Figure 16.- Time history solutions 2; orbit 2, $x(0)$ at -13° , $T = 400$ sec.

---- x_1 | DIV = 500 km
 ---- x_2 | DIV = 500 m/s
 -.- x_3 | DIV = 50 km
 — x_4 | DIV = 500 m/s

---- $r - r_c$ | DIV = 500 km
 ---- $-u_3$ | DIV = 25 kg/s
 -.- x | DIV = 500 km
 — y | DIV = 500 km



(a) STATES IN ROTATING
COORDINATE SYSTEM



(b) INERTIAL COORDINATES

Figure 18.- Time history solutions; orbit 3, $x(0)$ at -60° , $T = 1075$ sec.

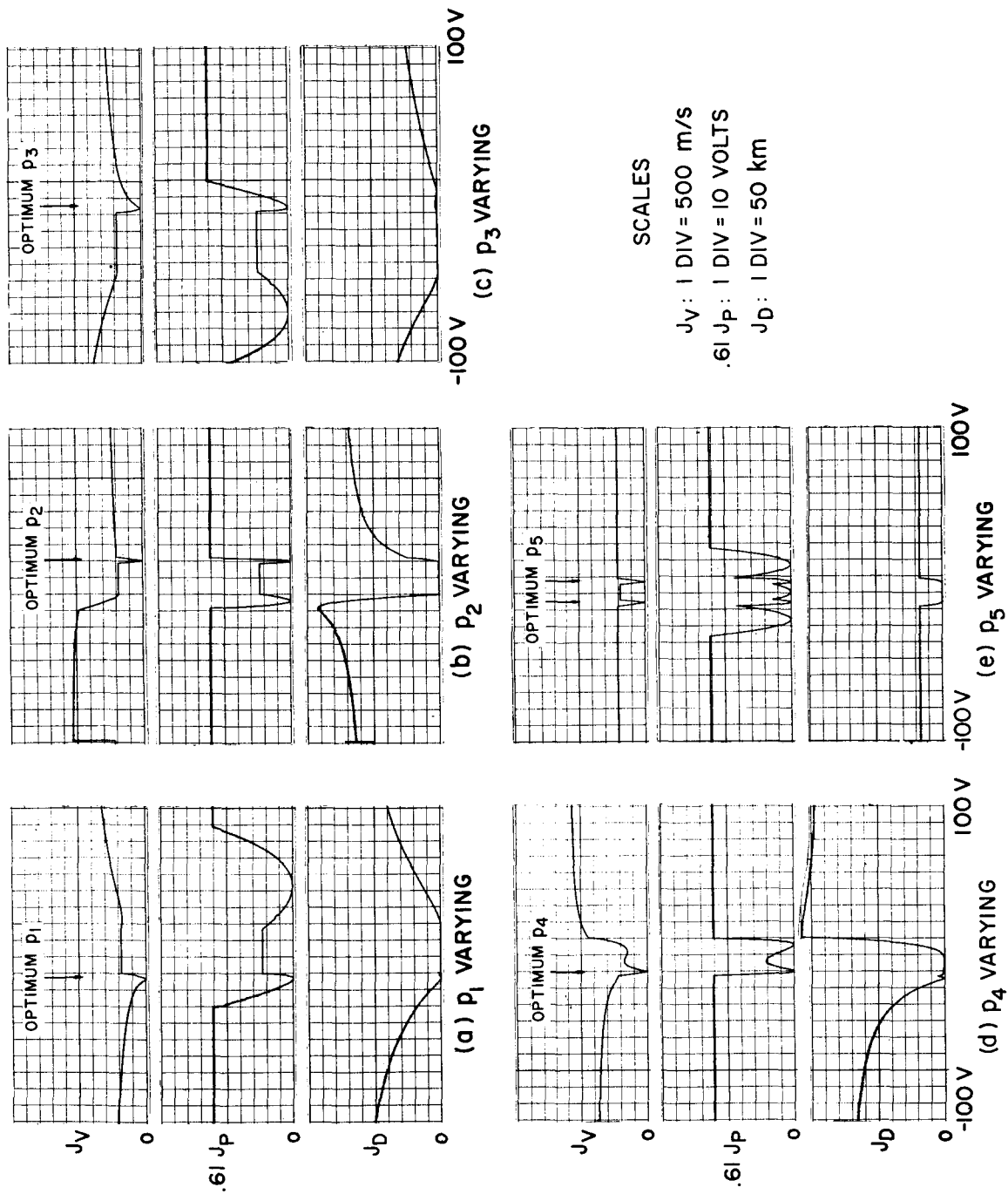


Figure 19.- Two-dimensional cross sections of boundary hypersurface; orbit 2, $x(0)$ at -37° , $T = 600$ sec.

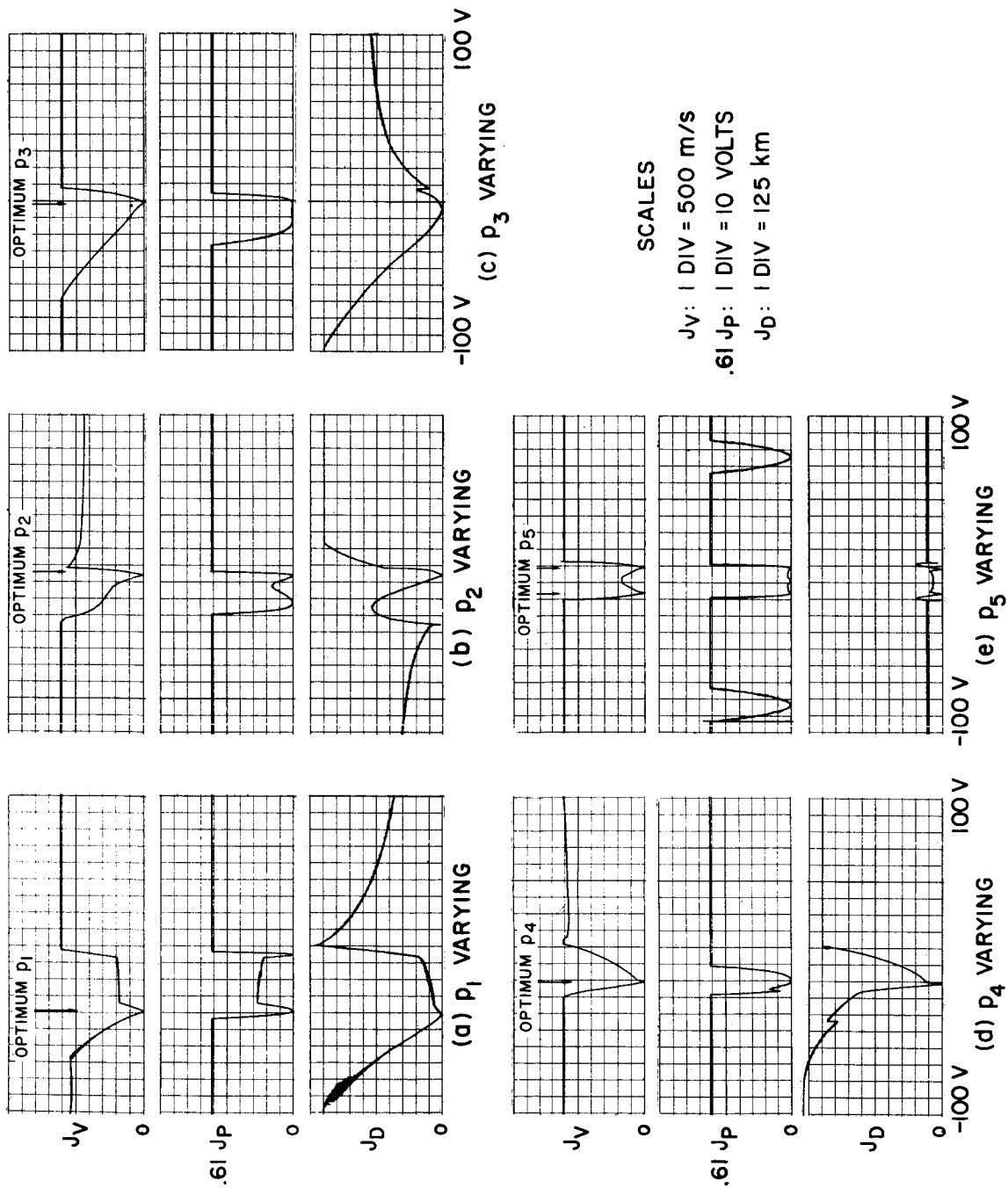
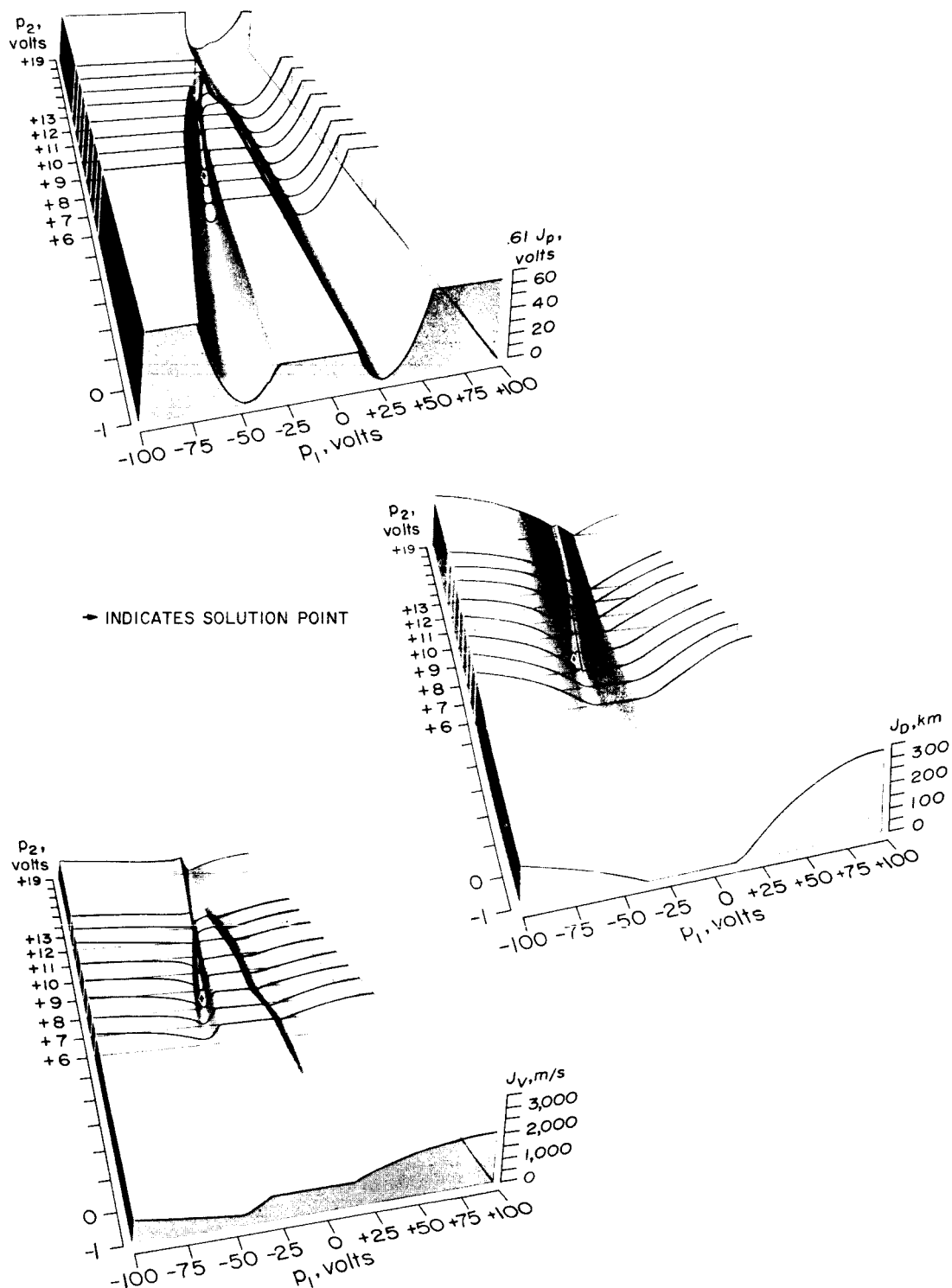
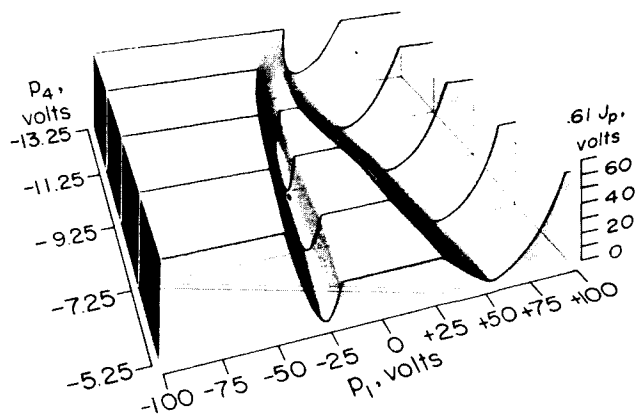


Figure 20.- Two-dimensional cross sections of boundary hypersurface; orbit 3, $x(0)$ at -60° , $T = 1075$ sec.

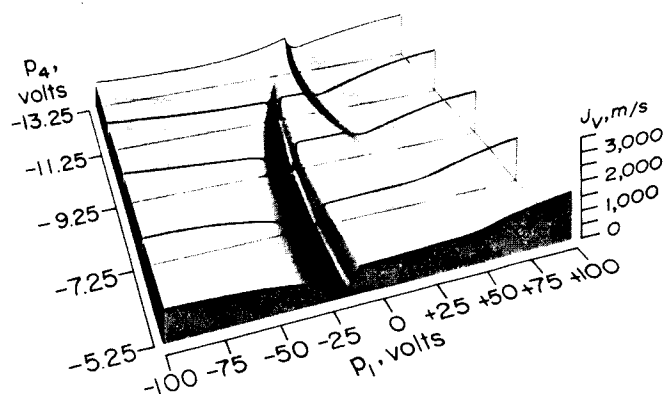
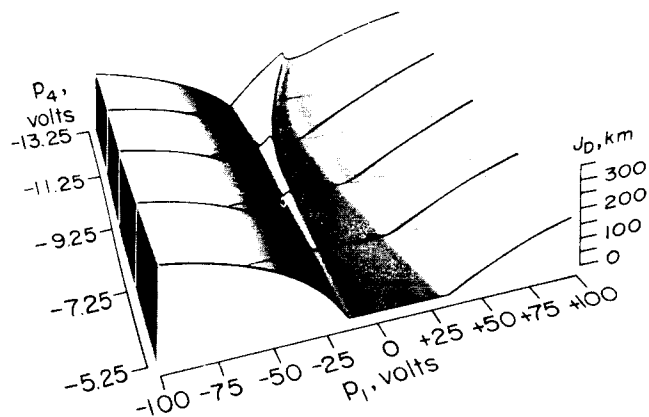


(a) p_1 , p_2 plane.

Figure 21.- Three-dimensional cross sections of boundary hypersurface; orbit 2, $x(0)$ at -37° , $T = 600$ sec.



→ INDICATES SOLUTION POINT



(b) P_1, P_4 plane.

Figure 21.- Concluded.